Algorithms

1) The sequence steps taken to solve a problem is called an algorithm.

2) Algorithm helps to create computer program for problem solving.
### Flowcharts

1) Graphical representation of the step by step solution of a problem is called flowchart.

2) Flowchart symbols and their purposes are given below:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Oval</td>
<td>Start or End of flowchart.</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Circle</td>
<td>Connector - Indicates flowchart continues where a matching name or number is found.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Arrow</td>
<td>Indicates flow direction.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Parallelogram</td>
<td>Input or output.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Rectangle</td>
<td>Processing</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Rhombus</td>
<td>Decision making (conditions - if/switch, loops - for/while)</td>
<td></td>
</tr>
</tbody>
</table>
Introduction to web servers

1 Web server sends HTML pages in response to request from client. Web server uses Hyper Text Transfer Protocol (HTTP).

2 Web server uses port number 80 for general data exchange and port number 443 for data exchange over Secure Socket Layer (SSL).

3 Scripts executed by the webserver are called server side scripts.

4 PHP, Perl, Python and ASP.NET are some of the programming languages used for server side scripting.

5 Some scripts embedded in HTML pages are executed by the browser. The scripts executed on browser are called client side scripts.

6 Javascript and VBScript are some of the most common client side scripting languages.

7 Microsoft Internet Information Server (IIS), Apache httpd, nginx, Google Web Sever (GWS) and IBM Websphere are some of the most common web servers.
Features of web server

1. Port numbers 80 and 443 are used by web servers.
2. Web address is called Uniform Resource Locator (URL).
3. URL parts are separated by forward slash (/) symbol.
4. If a URL begins with http:// the communication takes place on port number 80 and it is unsecure.
5. If a URL begins with https:// the communication takes place on port number 443. Secure Socket Layer (SSL) protocol is used to prevent data intrusion.
6. The last part of a web URL is the name of the file.
7. A folder named ‘htdocs’ or ‘wwwroot’ serves as the root directory of the web server.
Compiler based programming languages

1. The human readable program is called source code.
2. A special program called compiler converts the source code to machine readable binary code.
3. A source code should follow certain formatting details, called syntax, if the program should be successfully compiled.
4. Compiler detects errors and converts/ translates the code only if the entire program has correct syntax.
5. Since compilation is done only once, the time spent on removing syntax and logical errors (called debugging) does not affect the time required for running on a computer.
6. After compilation, machine code runs very fast on a computer.
7. Assembly language, C, C++, Java, C#, etc. are some of the most popular compiled languages.
8. Compiler based languages are very strict about syntax and data types.
Interpreter based programming languages

1. Interpreter is a special program to translate few lines of source code, load it to the processor based on program flow.
2. Interpreted programs are slower than compiled programs.
3. Interpreted programming languages enforce liberal syntax (they forgive some deviations) and have less stringent data types.
4. Interpreted programs are easier and faster to develop for small projects.
5. The liberal nature of interpreted programming languages permits more errors and more complications for large programs.
6. A class of languages suitable for embedding inside a larger program are called scripting languages.
7. Javascript, Perl, Python, Unix/Linux shell script, etc. are some of the most popular scripting languages.
Javascript

1 Javascript loosely follows the syntax of Java programming language.
2 Javascript is embedded inside HTML code.
3 A Javascript function is embedded inside a pair of `<script>` `<script>` tags.
4 The script tag begins with one or both of the following attributes: `type="text/javascript"` and `language="javascript"`.
5 Javascript can be used as a procedural language (only functions are permitted) or as an Object oriented language (new data types called classes are permitted).
6 Each line of Javascript code ends with a semi-colon (;), unless it begins a new block using curly brace({}).
7 Javascript runs inside a browser.
8 Script can be written within the HTML file itself or in an external file (with .js extension).
9 External script file (say welcome.js) may be embedded using `src` attribute of script tag.
Comments in Javascript

1. Comments in Javascript can cover single line or multiple lines.
2. Single line comments start with `//` symbol and run till the end of the line in which it is placed. For example,
   \[ x*=2; //\textit{current value of} \ x \ \textit{is doubled}. \]
   \[ //\textit{The following code calculates age of a person.} \]
   are examples of single line comment.
3. Multiple line comment starts with `/*` and ends only when `*/` is found. For example,
   \[ /* \]
   The following lines are ignored by the browser.
   \[ x+=5; \]
   \[ y=Math.pow(x,2); \]
   \[ */ \]
   is an example of multi-line comment.
5. Commented portions are ignored by the browser.
6. Comments help the programmer to understand the meaning of code at a later date.
Application of Javascript for web

1. Javascript can be embedded in HTML code.
2. Support for Javascript is available in all the modern web browsers.
3. Javascript methods can be easily invoked through onClick property of a button.
4. Javascript can be used to dynamically interact with the user of a web page.
5. Since Javascript is executed inside a browser, the danger of malware spreading to the system through the script is very low.
## Data types in Javascript

1. Javascript supports the following primitive (basic level) data types:

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Primitive type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>String</td>
<td>A sequence of characters is called a String. e.g., &quot;Hello&quot;. It is enclosed in double quotes.</td>
</tr>
<tr>
<td>2</td>
<td>Number</td>
<td>Any numerical value, whether integer or floating point is a Number. e.g. 3, 6.23.</td>
</tr>
<tr>
<td>3</td>
<td>Boolean</td>
<td>Contains the result of a logical expression. e.g., true, false.</td>
</tr>
<tr>
<td>4</td>
<td>Undefined</td>
<td>When a variable is undeclared, it is categorized as undefined.</td>
</tr>
<tr>
<td>5</td>
<td>NULL</td>
<td>It signifies the absence of any specific value.</td>
</tr>
</tbody>
</table>
Data types in Javascript

2. Javascript supports non-primitive data types called Object (reference type objects).

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Non-Primitive</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Object</td>
<td>Represents the basic reference type. New data types in Javascript may be defined using function itself.</td>
</tr>
<tr>
<td>2</td>
<td>Array</td>
<td>Array represents a collection of indexed values. var num=new Array(&quot;Zero&quot;, &quot;One&quot;, &quot;Two&quot;, &quot;Three&quot;); alert(num[2]); displays Two in the alert dialog.</td>
</tr>
<tr>
<td>3</td>
<td>RegExp</td>
<td>Regular expression used for manipulation of strings.</td>
</tr>
</tbody>
</table>
Variables and constants in Javascript

1. A variable represents a memory location in which values can be stored and retrieved.

2. Javascript variable names must start with alphabets (A-Z or a-z) or underscore (_ ) or dollar symbol ($).

3. Numbers can be used in variable names anywhere except the first location.

4. Variable declaration starts with the keyword `var`. e.g., `var $x=1,y=3, z;`

5. Javascript variables are case sensitive. i.e., `var a=10, A=20;` declares two different variables named `a` and `A`.

6. A variable declared outside the scope of any function is called a **global variable**.

7. If a declaration starts with the keyword `const` instead of `var`, the variable so declared can be assigned a value only once. e.g., `const PI=3.14159625;` will make `PI` a constant. New values assigned to it (like `PI=1;`) will fail.
Conversion between data types

1. Data of any type (e.g. "25.67") can be converted to number using the `Number()` function (e.g., `x=Number("25.67");`).

2. Data of any type can be converted to String using `String()` function (e.g. `str=String(23.23);`).

3. Data of any type can be converted to Boolean type (true or false) using `Boolean()` function.

4. The `typeof` operator helps to determine the data type of a variable. e.g.,
   ```javascript
   var name="Desiga", mark=95; alert(typeof name);
   alert(typeof mark);
   ```
   displays String and Number as the data types respectively.

5. `NaN` is a special number which indicates a value too large for storage in number type (like \( \infty \)).

6. `null` is another data type stored in a variable when there is no value in a variable.

7. `var x; alert(typeof x);` displays that `x` is undefined.
Arithmetic Operators in Javascript

1. Javascript provides operators for basic operations like Division, Multiplication, Addition, Subtraction, etc.

Table: List of arithmetic operators in Javascript

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Arithmetic Operator</th>
<th>Purpose</th>
<th>Example</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>**</td>
<td>Exponentiation</td>
<td>( x = 10 / 2; // x \text{ value is } 2. )</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>*</td>
<td>Multiplication</td>
<td>( x = 10 * 2; // x \text{ value is } 20. )</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>/</td>
<td>Division</td>
<td>( x = 10 / 2; // x \text{ value is } 2. )</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>%</td>
<td>Modulo Remainder</td>
<td>( x = 10 % 4; // x \text{ value is } 2. )</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>+</td>
<td>Addition</td>
<td>( x = 10 + 2; // x \text{ value is } 12. )</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>Subtraction</td>
<td>( x = 10 - 2; // x \text{ value is } 8. )</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>++</td>
<td>Increment</td>
<td>( x = 10; x + +; // x \text{ value is } 11. )</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>--</td>
<td>Decrement</td>
<td>( x = 10; x -- --; // x \text{ value is } 9. )</td>
<td>8</td>
</tr>
</tbody>
</table>
Assignment Operators in Javascript

1 Javascript provides assignment operators to assign the results of mathematical expression to variables.

Table: List of assignment operators in Javascript

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Assignment Operator</th>
<th>Purpose</th>
<th>Example</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>=</td>
<td>Assignment</td>
<td>$x = 10$; // $x$ value is 10.</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>+=</td>
<td>Add and Assign</td>
<td>$x = 10; x += 4;$ // $x$ value is 14.</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>-=</td>
<td>Subtract and Assign</td>
<td>$x = 10; x -= 4;$ // $x$ value is 6.</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>*=</td>
<td>Multiply and Assign</td>
<td>$x = 10; x *= 4;$ // $x$ value is 40.</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>**=</td>
<td>Take power and Assign</td>
<td>$x = 2; x **= 4;$ // $x$ value is 16 ($2^4 = 2 \times 2 \times 2 \times 2 = 16$).</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>/=</td>
<td>Divide and Assign</td>
<td>$x = 10; x /= 4;$ // $x$ value is 2.5</td>
<td>6</td>
</tr>
</tbody>
</table>
Comparison operators in Javascript

Comparison operators compare the left and right side results based on Boolean logic (true or false) results. List of comparison operators is shown in Table 3.

Table: Comparison operators in Javascript

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Operator</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>==</td>
<td>Equal to (ignore data type. e.g., &quot;20&quot;==20 gives true as the result, since data types are not considered.)</td>
</tr>
<tr>
<td>2</td>
<td>===</td>
<td>Equal to (compare data type and value. &quot;20&quot; === 20 is false.)</td>
</tr>
<tr>
<td>3</td>
<td>!==</td>
<td>Not equal to. Ignore data type.</td>
</tr>
<tr>
<td>4</td>
<td>!===</td>
<td>Not equal to. Gives true result only when the data type and value are identical.</td>
</tr>
<tr>
<td>5</td>
<td>&gt;</td>
<td>Greater than.</td>
</tr>
<tr>
<td>6</td>
<td>&gt;=</td>
<td>Greater than or equal to.</td>
</tr>
<tr>
<td>7</td>
<td>&lt;</td>
<td>Less than.</td>
</tr>
</tbody>
</table>
Logical operators in Javascript

1 Logical operators help to combine the results of several logical operations (AND, OR, NOT, etc.) based on Boolean logic. Table 4 shows the logical operators used in Javascript.

Table: Logical operators in Javascript

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Operator</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&amp;&amp;</td>
<td>Logical AND.</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>!</td>
<td>Logical NOT operator (Negation).</td>
</tr>
</tbody>
</table>
Special operators in Javascript

1. Following are some of the uncategorized, but important operators used in Javascript (Table 5).

Table: Special operators in Javascript

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Operator</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>typeof</td>
<td>Returns the data type of a variable.</td>
</tr>
<tr>
<td>2</td>
<td>new</td>
<td>Creates a new object.</td>
</tr>
<tr>
<td>3</td>
<td>? : ternary</td>
<td>Evaluates the logical expression to the left of ?. Executes left side of : if it is true, right side of : otherwise. e.g., <code>var x = 2; y = (x! = 2)?5 : 8;</code> sets y value to 8, since the logical expression is false.</td>
</tr>
<tr>
<td>4</td>
<td>instanceof</td>
<td>Checks whether a variable belongs to particular data type.</td>
</tr>
<tr>
<td>5</td>
<td>in</td>
<td>Takes a value from an array. Used in for loops. e.g., <code>var x</code></td>
</tr>
</tbody>
</table>
String operators in Javascript

1 String class supports assignment and string concatenation operators (Table 6).

Table: String operators in Javascript

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Operator</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>=</td>
<td>Assigns string value on the right side to a variable on the left side. e.g., var name=&quot;Gayathri&quot;.</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>Adds 2 strings together. var x=&quot;COPA&quot;, y=&quot;is great.&quot;; var z=x+y; sets &quot;COPA is great.&quot; as the value in z.</td>
</tr>
<tr>
<td>3</td>
<td>+=</td>
<td>Appends the string on right side to the variable on left side. e.g., var x=&quot;Sun&quot;, y=&quot;is the nearest star.&quot;; x+=y; sets &quot;Sun is the nearest star.&quot; in variable x.</td>
</tr>
</tbody>
</table>
Arrays in Javascript

1. Array is a collection of values in a single variable.
2. Array elements are numbered from 0 (0, 1, 2, 3, ...).
3. Each element of an array may be accessed using []. e.g. 
   \[x = \{3, 5, 2, 10\}; \text{alert}(x[1]);\] displays 5.
4. In a for loop, elements of array may be accessed using index values (0 to n) or using \textit{in} operator. e.g.,
   \[x = \{3, 5, 2, 10\};\]
   \[\text{for(var } i=0; i<x.length; i++) \text{alert}(x[i]);\]
   and
   \[x = \{3, 5, 2, 10\};\]
   \[\text{for(var } y \text{ in } x \text{)} \text{alert}(x[y]);\]
   display all the elements array \(x\).
5. Array may be declared using a pair of \{ \} or \texttt{new Array()}; function.
Associative arrays in Javascript

1 Instead of accessing array elements using a numeric index, it is possible to permit some name to each element.

2 Such an array having a name for index is called an associative array or hash map.

3 An associative array may be declared in any one of the following styles:
   
   ```javascript
   var x = {'Tamil':92, 'English':80, 'Maths':83, 'Science':87, 'Social Science':81};
   ```
   
   or
   
   ```javascript
   var x = new Array('Tamil':92, 'English':80, 'Maths':83, 'Science':87, 'Social Science':81);
   ```

4 Left side of an associative array or hash is called key and the right side is called value. e.g., in array x, 'Maths' is key and 83 is value.

5 Elements of associative arrays are accessed using the key. e.g.,
   
   ```javascript
   y = x['Tamil'];
   ```
   sets value of y to 92.
Control statements in Javascript

1. Control statements help in conditional execution of portions of code.

2. Conditional controls may be imposed using
   (i) if...else condition,
   (ii) switch ... case ... break condition, or
   (iii) ternary operator (?:).
If ... else condition

1. The *if* condition contains a condition part. In case the condition is true, the block (enclosed by a pair of `{ }`) immediately following *if* condition is executed.

2. An *if* condition having a single line might omit the pair of `{ }`.

3. In case the *if* condition fails, an optional *else* block may be coded. *Else* may also check for a condition using an *if* (*else if*).

4. *Else* block may also omit the pair of `{ }` if the code for it contains a single line.
Example of if condition

```html
<html>
<head>
<title>Test of if condition</title>
<script type="text/javascript" language="javascript">
function ageMessage() {
  var age = this.ageForm.age.value;
  if(age < 13) {
    document.write("<h1 style='color:#FF0000'>You are a kid</h1>" );
  }
  else if(age < 20) {
    document.write("<h1 style='color:#FF0000'>You are a teenager!</h1>" );
  }
  else if(age < 36)
    document.write("<h1 style='color:#FF0000'>You are young!</h1>" );
  else if(age < 50)
    document.write("<h1 style='color:#FF0000'>You are middle aged!</h1>" );
  else
    document.write("<h1 style='color:#FF0000'>You are old!</h1>" );
}
</script>
</head>
<body>
<form name="ageForm">
<center>
Enter your age: <input type="number" name="age"/></center>
<input type="button" value="Get message" onClick="ageMessage()"/>
</form>
</body>
</html>
```
How the if condition program works?

1. The if ladder first tests whether the age is less than 13. In case it is correct, the message "You are a kid!" is displayed.

2. In case the value is the first test fails (age ≤ 13), the else if part tests whether the age is between 13 and 19. In such a case, the message "You are a teenager!" loads.

3. In case the age is greater than or equal to 20, the third else if condition checks whether the age is less than 36. If so, the message "You are young!" is displayed.

4. In case the age is greater than or equal to 36, the third else if condition checks whether the age is less than 50. If so, the message "You are middle aged!" is displayed.

5. When all the if conditions fail, the last else is reached. It displays the message "You are old!".
Switch condition

1 Switch condition takes one argument (e.g. switch(x)).
2 It checks the input against several case values (e.g., case 2:) and executes the lines following the matching case.
3 All cases, other than the last one end with break keyword.
4 The last condition may be called default :, which handles cases not matching any of the previous ones.
Test of switch condition

```html
<html>
<head>
<title>Test of switch condition</title>
<script type="text/javascript" language="javascript">
function nameMessage() {
  var name = this.nameForm.name.value;
  var length = name.length;
  switch(length) {
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
      document.write("<h1 style='color:#FF0000'>Very short name! </h1>"');
      break;
    case 6:
    case 7:
    case 8:
    case 9:
    case 10:
      document.write("<h1 style='color:#FF0000'>Length of your name is normal! </h1>"');
      break;
    case 11:
      document.write("<h1 style='color:#FF0000'>Your name is bit long! </h1>"');
      break;
    default:
      document.write("<h1 style='color:#FF0000'>Your name is too long! </h1>"');
  }
}
</script>
</head>
<body>
<form name="nameForm">
<center>
Enter your name: <input type="text" name="name" /></br />
<input type="button" value="Get message" onClick="nameMessage()" />
</center>
</form>
</body>
</html>
```
How the switch...case program works?

1. The switch takes length of name as argument.
2. For cases 1, 2, 3, 4 and 5, it displays the message "Very short name!".
3. For cases 6, 7, 8, 9 and 10, it displays the message "Length of your name is normal!".
4. For case 11, it displays the message "Your name is bit long!".
5. As the default case (length > 11), it displays the message "Your name is too long!".
6. Note that cases and case groups end with break.
7. The last condition (default) need not have a break.
The ternary operator (?:)

1. A simple if ... else condition may be re-written using one liner called ternary.

2. Ternary operator has 3 parts.

3. First part of the ternary operator (to the left of ?) is the condition. The condition may be either true or false.

4. The second part of the ternary, placed to the right of ? and the left of :, is executed if the condition is true.

4. The third part of the ternary, placed to the right of :, is executed if the condition is false.

5. Consider the following lines:
   
   ```
   x = 25;
   y = (x < 21) ? "Good" : "Bad";
   ```
   
   can you guess the value of y?

6. y has the value of “Bad”, since the condition turns out to be false.
Example of ternary operator (?:)

```html
<html>
<head>
<title>Test of if condition</title>
<script type="text/javascript" language="javascript">
function bmiMessage() {
    var wt = this.bmiForm.wt.value,
        ht = this.bmiForm.ht.value/100; //Convert height from centimeter to metre.
    var bmi = wt/ht/ht;
    var mess = "Your BMI is ";
    mess += bmi <= 20.3 ? "very good!" : "very high. You need exercise!"
    document.getElementById("res").innerHTML = "<h1 color=#FF0000>" + mess + "</h1>";
}
</script>
</head>
<body>
<form name="bmiForm">
<center>
Enter weight: <input type="number" name="wt" /></br />
Enter height in centi metre: <input type="number" name="ht" /></br />
<input type="button" value="Get message" onClick="bmiMessage()" />
</center>
</form>
</body>
</html>
```
Loops in Javascript

1 Javascript supports 4 types of loops:
   (a) *for* loop with index value.
   (b) *for* — *in* loop with list of array elements.
   (c) *while* loop.
   (d) *do* — *while* loop.

2 For loop having index value handle only integer index.
1 Index value of *for* loop may be declared before commencement of the loop or at the loop itself. (e.g., *var i; for*(*i* = 0; *i* < 10; *i*++) is equivalent to *for*(var *i* = 0; *i* < 10; *i*++)).

2 The *for* loop has 3 parts, viz., (i) initialization part, (ii) condition part, (iii) increment part.

3 The initialization part sets the initial value of index variable.

4 The *for* loop keeps running so long as the condition part (second of the three parts) is true.

5 The increment part increases or decreases value of variable. e.g., *for*(var *i* = 0; *i* < 100; *i*++), *for*(var *i* = 0; *i* < 100; *i*++ = 2), *for*(var *i* = 100; *i* >= 0; *i* --), *for*(var *i* = 0; *i* < 100; ++*i*), *for*(var *i* = 0; *i* < 100; *i*++ = 10) are all valid increments to the index variable of *for* loop.

6 Each of the 3 parts of *for* loop is separated by a ; (semi-colon).

7 One of more of the parts of *for* loop may be omitted. e.g., var *i* = 0; *for*(; *i* < 10; *i*++), *for*(var *i* = 0; *i* < 10; ).
Example of for loop

```html
<html>
<head>
<title>Example of for loop</title>
</head>
<body>
<script type="text/javascript" language="javascript">  
var x = new Array(4,2,6,8,12,15,25), sum=0;
  document.write("Sum of ");
for(var i=0; i<x.length; i++) {
    document.write(x[i] + (i===x.length-1?" = ":" + ");
    sum += x[i];
}
  document.write(sum);
</script>
</body>
</html>
```
for — in loop in Javascript

1. *for — in* loop is used in conjunction with arrays and lists.

2. A typical *for — in* loop to move through the elements of an array creates an index variable which changes value from the first index to the last.

3. The *for — in* loop is a convenient replacement to *for* loop having separate declaration, index limit checking and increment parts.
Example of *for* — *in* loop

```html
<html>
<head>
<title>Example of for-in loop</title>
</head>
<body>
<script type="text/javascript" language="javascript">
var x = new Array(4,2,6,8,12,15,25), sum=0;
document.write("Sum of ");
for(var y in x) {
    document.write(x[y] + " ");
    sum += x[y];
}
document.write(" is "+sum);
</script>
</body>
</html>
```
Mark statement using *for—in* loop and associative array

```html
<html>
<head>
<title>Example of for-in loop</title>
</head>
<body>
<script type="text/javascript" language="javascript">
var x = {'Tamil':92, 'English':98, 'Maths':100, 'Science':94, 'Social':91};
var sum=0;
document.write("<center><h1>Mark statement</h1> <table border=1>");
document.write("<tr><th>Subject</th> <th>Mark</th></tr>");
for(var y in x) {
    document.write("<tr><td>"+y+"</td><td style='text-align:right'>"+x[y]+"</td></tr>");
    sum += x[y];
}
document.write("<tr><td style='color:#FF0000; text-align:center'>Total</td><td style='color:#FF0000; text-align:right'>"+sum+"</td></tr></table>");
</script>
</body>
</html>
```
While loop in Javascript

1. While loop checks a condition. If the condition yields *true*, the loop portion is executed.

2. If the loop contains only one statement, use of a pair of curly braces (`{ }`) is optional.

3. If the loop contains more than one statement, the entire loop body is enclosed in a pair of curly braces (`{ }`).

4. Index variable of the while loop should be declared and initialized before the while loop begins.

5. Index variable, if any, should be incremented or decremented in a separate line within the while loop.
Example of while loop

```html
<html>
<head>
<title>Example of for-in loop</title>
<script>
function table() {
  var x=this.document.tabForm.n.value, y, i=1;
  document.write("<center><h1 style='color:#FF0000'>" +x+" Table</h1><h2>"");
  while(i<=10) {
    y = i*x;
    document.write(i+" x " +x+" = " +y+"<br />");
    i++
  }
  document.write("</h2></center>"
}
</script>
</head>
<body>
<center>
<form name="tabForm" method="GET" action="#">
Enter table number: <input type="number" name="n" />
<input type="button" value="Create table" onClick="table()" />
</form>
</center>
</body>
</html>
```
The do ... while loop in Javascript

1. Do ... while loop is a special loop which gets executed at least once, because the condition is checked only at the last statement of the loop.
2. The index variable, if any, should be declared outside the loop.
3. The do ... while loop checks the condition at the last line (which is a while).
Example of do ... while loop - Fibonacci series

<html>
<head>
<title>Example of do ... while loop</title>
</head>
<body>
<script>
var i=1, x=1,y=1;
document.write("""
<h1 style='color:#FF0000'>Fibonacci series</h1>
""");
document.write("""
<h1 style='color:#0000FF'>"+(i++)+" +x+"<br /></h1>
""");
var next=false;
do {
document.write((++i)+" +y+"<br />
);
var z=x;
x=y;
y+=z;
next = confirm("Do you want the next Fibonacci number");
}while(next);
document.write("""
</h2>
""");
</script>
</body>
</html>
Functions in Javascript

1 Generally, functions have 3 components:
   (a) The function keyword at the beginning.
   (b) The name of function following the function keyword. The name of function contains pair of brackets (), which might contain arguments.
   (c) Body of the function enclosed in a pair of curly braces ({}). The function might use *return* keyword to return a value after processing data.

2 In general, a named block of code is called function, which may be invoked using the given name.

3 A variable declared inside the function dies within that function.

4 A function may access variables declared inside the function or with a global scope (outside any function).

5 Any changes made by a function to the global variables is permanent and is visible in other functions too.
The following code shows 3 functions, having different combination of arguments and return values.  

```html
<html>
<head>
<title>
Test of Javascript functions
</title>
<script>
//Function without argument and without return value
function add1() {
var x = prompt("Enter x"), y = prompt("Enter y"),
z=Number(x)+Number(y);
alert(x+" + "+y+" = "+z);
}

//Function with arguments but without return value
function add2(x, y, z) {
var z=Number(x)+Number(y);
alert(x+" + "+y+" = "+z);
}

//Function with arguments and return value
function add3(x, y) {
var z = Number(x)+Number(y);
return z;
}
</script>
</head>
<body>
<script>
add1();
add2(5, 10);
var res = add3(20,45);
document.write("20 + 45 = "+res);
</script>
</body>
</html>
```
Use of function for OOPS in Javascript

1. OOPS stands for Object Oriented Programming System.
2. Object is a programming entity having properties (to store values) and methods (to work with properties).
3. Javascript supports Object creation using function.
4. A function can be converted to a class definition.
5. Variable declared inside the function is accessed with dot operator.
6. A method is declared using the expression `this.method = function() { }`.
Example of objects using Javascript function

```html
<html>
<head>
<title>OOPS using Javascript</title>
<script>
function Student(name, dob, grade) {
    this.name = name;
    this.dob = dob;
    this.grade = grade;
    this.getName = function() {
        return this.name;
    }
    this.getDOB = function() {
        return this.dob;
    }
    this.getGrade = function() {
        return this.grade;
    }
    this.setGrade = function(gr) {
        this.grade = gr;
    }
}

var s1 = new Student('Desiga', '2000-05-12', 'A'),
    s2 = new Student('Gayathri', '2001-03-01', 'B');

document.write("<h1 style='color:#FF0000'>Before changing grade</h1><h2 style='color:#0000FF'>" +
    s1.getName() + " : " + s1.getDOB() + " : " + s1.getGrade() + " <br /></h2>");

document.write(s2.getName() + " : " + s2.getDOB() + " : " + s2.getGrade() + " <br />");

s1.setGrade('S');

s2.setGrade('A');
document.write("<h1 style='color:#FF0000'>After changing grade</h1><h2 style='color:#0000FF'>" +
    s1.getName() + " : " + s1.getDOB() + " : " + s1.getGrade() + " <br /></h2>");

document.write(s2.getName() + " : " + s2.getDOB() + " : " + s2.getGrade() + " <br />");
</script>
</head>
<body>
</body>
</html>
```
Built in JavaScript functions overview

1. Functions which are readily available in the general environment of Javascript are called built-in functions.

2. Built-in functions help to process data belonging to the built-in data types.

3. Built-in functions of Javascript may be categorized under the following heads:
   (a) Number functions
   (b) String functions
   (c) Array functions
   (d) Date functions
   (e) Math functions

4. The built-in functions belonging to the above categories are described in the following sections.
Built-in functions in Number data type

- **toExponential()**: Formats a number in exponential ($X \times 10^y$) notation.
- **toFixed(digits)**: Formats a number with given number of digits after the decimal point.
- **toLocaleString()**: Formats a number based on browser geo-location.
- **toPrecision(digits)**: Formats the number with given total number of digits (including those to the left and right of decimal point).
- **toString()**: Returns String value.
- **valueOf()**: Returns value of the number.
Built-in functions in String data type

- `charAt(index)`: Returns character at given index location. Counting starts from 0.
- `charCodeAt(index)`: Returns UNICODE value of the character at given location. Counting starts from 0.
- `concat(s1,s2, s3)`: Joins given Strings (s1, s2, s3, etc.) with original String and returns the joined String.
- `indexOf(s1)`: Returns the location where the given String s1 occurs inside the original String. Returns -1 if no occurrence of s1 is available inside the original String.
- `lastIndexOf(s1)`: Returns the last location where the given String s1 occurs inside the original String. Returns -1 if no occurrence of s1 is available inside the original String.
- `localeCompare(s1)`: Compares original String and given String. Returns -1 if the original String sorts before given String, 0 if both are equal and 1 if the original String sorts after the given String.
- `length()`: Returns the length of String.
Built-in functions in String data type

- **match(s1)**: Compares the original and given Strings based on regular expression notation.
- **replace(s1, s2)**: Search for occurrences of s1 inside the original String and replace with s2. String s1 may have regular expression.
- **search(s1)**: Searches the original String for the given String s1. The String s1 may have regular expression.
- **slice(begin, end)**: Splits the original String between given beginning and end points. Returns the split String.
- **toLowerCase() / toUpperCase()**: Returns lower case version (or upper case version) according locale String.
- **toLocalLowerCase() / toLocalUpperCase()**: Returns lower case version (or upper case version) according locale String based on current geo-locale.
Built-in functions in Array data type

- **concat(a1, a2, a3)**: Appends arrays a1, a2, a3, etc. to the original array and return the resulting array.
- **every(function)**: Returns values which satisfy given function. The function should return Boolean value.
- **filter(function)**: Returns values which satisfy given filter function. The function should return Boolean value.
- **forEach(function)**: Calls given function for each element of the array.
- **indexOf(val)**: Returns the first occurrence location of the given value in the original array.
- **join()**: Creates a String using the values of the array.
- **lastIndexOf(val)**: Returns the last location at which given value occurs.
- **map(function)**: Calls given function with each element of the array and returns the results in new array.
Built-in functions in Array data type

- **pop()**: Remove the last element of the array.
- **push()**: Add new element to the end of the array.
- **reduce()**: Apply given function to the right and left elements and replace them with given result till there remains only a single element.
- **reduceRight()**: Reduces given array from right to left, applies given function and returns a single value.
- **shift()**: Removes the first element of the array. Returns the value of removed element.
- **slice(begin, end)**: Splits the array from given beginning through the given end.
- **sort()**: Sorts elements of the array.
- **splice(location, length, value)**: Adds given value at given location. It removes elements at given location if the length is greater than 0.
- **unshift()**: Adds one element to the beginning of array.
Built-in functions in Date data type

- **Date()**: Date and time of today and now.
- **getDate()**: Returns day of month (month counting starts at 0).
- **getDay()**: Returns day of week (day count starts at 0).
- **getFullYear()**: Returns year stored in the date object in 4 digit format.
- **getHours()**: Return the time stored in the date object in 24 hour format.
- **getMilliseconds()**: Returns the millisecond value of time stored in date object.
- **getMinutes()**: Returns the minutes part of time stored in the date object.
- **getMonth()**: Returns month value stored in the date object (month count starts at 0).
- **getSeconds()**: Returns the seconds part stored in the date object.
Built-in functions in Date data type

- `getTime()`: Returns milliseconds passed since the epoch date of 01.01.1970 (midnight).
- `getUTCDate()`, `getUTCDay()`, `getUTCFullYear()`, `getUTCHours()`, `getUTCMilliseconds()`, `getUTCMinutes()`, `getUTCMonth()`, `getUTCSeparators()`:
  get day of month, day of week, year, hours, milliseconds, minutes, month and seconds in Coordinated Universal Time.
- `setDate(dt)`: Set `dt` value in the day of month part of date object.
- `setFullYear(y)`: Set `y` value in full year part of date object.
- `toDateString()`: Returns the date and time values in human readable string format.
- `toGMTString()`: Returns human readable date and time in Greenwich Mean Time format.
- `toLocalFormat()`: Converts date to given format string.
- `toLocaleString()`: Returns string value of date to the local date and time format.
Built-in functions in Date data type

- `toLocaleTimeString()`: Returns time part of the date according to local format.
- `toSource()`: Return string equivalent of date object.
- `toUTCString()`: Returns string value of date object in Coordinated Universal Time format.
- `valueOf()`: Returns the primitive value of date object.
- `Date.parse(str)`: Parse the string and obtain equivalent date object.
- `Date.UTC()`: Returns milliseconds value based on Coordinated Universal Time.
Built-in functions in String Math type

- **Math.abs(x)**: Returns absolute value of given number, ignoring + or - signs.
- **Math.acos(x)**: Returns arc cosine \((\cos^{-1})\) value of \(x\).
- **Math.asin(x)**: Returns arc sine \((\sin^{-1})\) value of \(x\).
- **Math.atan(x)**: Returns arc tangent \((\tan^{-1})\) value of \(x\).
- **Math.atan2(x, y)**: Returns the arc tangent \((\tan^{-1})\) of \(\frac{x}{y}\) value.
- **Math.ceil(x)**: Returns the integer just greater than given fraction \(x\).
- **Math.cos(x)**: Returns the cosine value of \(x\).
- **Math.exp(x)**: Return \(e^x\) value of \(x\), where \(e\) is the base for Napier’s logarithm \((e = 2.718281828459)\)
- **Math.floor(x)**: Returns the integer just lower than the given fraction \(x\).
- **Math.log(x)**: Returns natural logarithm to the base \(e\) for the value of given \(x\).
Built-in functions in Math data type

- **Math.log10(x)**: Returns logarithm to the base 10 for value of x.
- **Math.max(x,y,z,...)**: Returns maximum of the given set of values.
- **Math.min(x,y,z,...)**: Returns minimum of the given set of values.
- **Math.pow(x,y)**: Returns $x^y$ value.
- **Math.random()**: Returns random number between 0 and 1.
- **Math.round(x)**: Rounds given number x to the nearest integer value.
- **Math.sin(x)**: Returns sine value of x.
- **Math.sqrt(x)**: Returns square root of x.
- **Math.tan(x)**: Returns tangent value of x.
Built-in functions in RegExp data type

- `RegExp.match(exp)`: Matches given `exp` regular expression against the given string. Returns the matching part.
- `RegExp.test(exp)`: Matches given `exp` regular expression against the given string. Returns true if a matching part is found.
Popup boxes in Javascript

1 Popup box opens out of a web page to convey an information, to confirm an action or to obtain an input.

2 Following types of popup boxes are supported by Javascript:
   
   (i) **alert**(message): Displays given message through a popup dialog.
   
   (ii) **confirm**(message): Confirms whether given option should be **OK** or **Cancel**.
   
   (iii) **prompt**(message, default): Prompts for an input by displaying a message and a default value (optional).
Example of dialogs in Javascript

```html
<html>
<head>
<title>Dialogs in Javascript</title>
</head>
<body>
<script>
var x = prompt("Enter x:","10");
var y = prompt("Enter y:","20");
if(confirm("Do you wish to add the numbers?")) {
    var z = Number(x)+Number(y);
    alert(x+" + "+y+" = " +z);
}
else
    alert("You have chosen not to do anything with the given numbers");
</script>
</body>
</html>
```
Document Object Model (DOM)

1. Accessing elements of a web page as though they belong to a document (using the this.document interface methods and properties) is called Document Object Model (DOM).
2. Document Object Model (DOM) may be visualized as shown in Fig. 1.

Figure: Document Object Model[1]
Properties of Document Object Model (DOM)

3 DOM supports access to HTML elements. Such element properties of DOM are shown in Fig.2.

Figure: Properties of Document Object Model[2]
Methods and properties of Document Object Model (DOM)

(1) `getElementById(id)`: Returns the element having the given id from the HTML document.

(2) `getElementByName(name)`: Returns the HTML element having the given name.

(3) `getElementsByTagName(tag)`: Returns an array of elements having the given tag name.

(4) `getElementsByClassName(cls)`: Returns an array of objects belonging to given class name.

(5) `write(str)`: Writes given string value to the document.

(6) `writeln(str)`: Writes given string value to the document and appends a newline character.

(7) `innerHTML`: Sets HTML portion for a particular element of the document.

(8) `innerText`: Sets text value for a particular element of the document.
Example of DOM

```html
<html>
<head>
<title>
Test of DOM
</title>
<script type="text/javascript">
function welcome() {
var name = document.nameForm.n.value; // DOM access
document.write("Welcome "+name+"!"); // DOM method write is used
}
</script>
</head>
<body>
<form name="nameForm" method="GET" action="#">
<center>
Enter name: <input type="text" name="n"><br />
<input type="button" value="Message" onClick="welcome()">
</center>
</form>
</body>
</html>
```
Browser Object Model (BOM) in Javascript

1. Browser Object Model (BOM) is supported by the window class.
2. The window class of BOM is the parent of all other objects. The structure of BOM is shown in Fig.3.

Figure: Properties of Browser Object Model[2]
Methods of window object in BOM

1. `alert(msg)`: Displays alert message.
2. `prompt(msg, default)`: Displays input dialog with given message and default value.
3. `confirm(msg)`: Displays confirmation dialog with OK and Cancel options.
4. `open()`: Opens a new window.
5. `close()`: Closes current window.
6. `setTimeout()`: Waits for given amount of time before taking action.
Methods of window.history object in BOM

1. `window.history.forward()`: Equivalent to pressing forward button in browser.
2. `window.history.back()`: Equivalent to pressing back button in browser.
3. `window.history.go(number)`: Loads given page number in history object.
Methods of window.history object in BOM

1. `window.screen.width`: Returns the width of screen.
2. `window.screen.height`: Returns the height of screen.
3. `window.screen.availWidth`: Returns available width after deducting the width of unusable portions.
4. `window.screen.availHeight`: Returns available height after deducting the height of unusable portions (like title bar, status bar, etc.).
5. `window.screen.colorDepth`: Returns color depth.
6. `window.screen.pixelDepth`: Returns pixel depth.
Introduction to VBA features and applications. VBA Data types, Variables and Constants. Operators in VBA and operator precedence. Mathematical Expressions in VBA. Introduction to Strings in VBA. Introduction to Arrays in VBA. Conditional processing in VBA, using the IF, Elseif, Select ... Case Statements. Loops in VBA. VBA message boxes and input boxes. Introduction to Creating functions and Procedures in VBA. Using the built in functions. Creating and editing macros. Introduction to Object Oriented Programming Concepts. Concepts of Classes, Objects, Properties and Methods. Events and Event driven programming concepts. The user forms and control in Excel VBA. Properties, events and methods associated with the Button, Check Box, Label, ComboBox, Group Box, Option Button, List Box, Scroll Bar and Spin button controls. Overview of ActiveX Data objects. Debugging Techniques.
Features of VBA

1. VBA stands for Visual Basic for Applications.
2. VBA provides a powerful programming environment within MS Office.
3. VBA can manipulate data in Excel spreadsheets.
4. VBA can be used to create new user interfaces using buttons, labels, text boxes, combo boxes, radio buttons and lists.
5. VBA is capable of Object Oriented Programming System (OOPS).
6. VBA programs cannot run as standalone programs. VBA code requires a “Host Application” (like MS Excel, MS Word, MS Powerpoint, etc.) to run.
Features of VBA

7 VBA provides the ability to automate repeated works carried out using MS Excel.
8 VBA source code is compiled into Microsoft P-Code (packed code).
9 Microsoft P-code is saved inside an MS Office file (.docx, .xlsx, .pptx, etc.).
10 MS Office applications use a hypothetical computer processor called Virtual Machine, to run the P-code of VBA.
11 The virtual machine communicates with the processor and gets the work executed.
12 Comment in VBA begins with a single quote (‘) and continues till the end of that line.
13 VBA variables can be declared on the fly. If an unknown variable is used, VBA assumes that it is newly declared.
14 On the fly declaration of variable may lead to errors, since a mistyped value (like tset instead of test) would be taken as a new variable.
15 If the sentence `Option Explicit` is included at the beginning of source code, any undeclared variable will result in an error message.
Applications of VBA

1. If a text string needs to be inserted repeatedly at many places in a document, VBA script will simplify the task.
2. VBA can be used to automate repeated processes required to create a report from available data.
3. VBA can be used to check errors or wrong entries in spreadsheet, since it has sophisticated access system to handle cells.
4. VBA can be used to create a small front end, which will insert data to a spreadsheet in the background. This will prevent erroneous entry of data.
5. When same set of operations are to be performed on several Excel sheets, VBA will help to automate the task.
Applications of VBA

6 When a single keystroke or button is needed to carry out a set of operations, VBA can be used to create that keystroke or button.

7 VBA can be used to create a custom command on the menu bar.

8 VBA can be used to create a custom button on any tool bar.

9 VBA can be used to create custom plugins for Excel.

10 VBA can be used to create a complete application, with the only limitation that it will be running inside MS Excel.
Data types in VBA

Following are the data types supported by VBA (Table 7):

Table: Data types in VBA

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Data type</th>
<th>Size (Byte)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Boolean</td>
<td>2</td>
<td>Stores \textit{true} (0) or \textit{false} (-1) value.</td>
</tr>
<tr>
<td>2</td>
<td>Byte</td>
<td>1</td>
<td>Stores integer values 0 to 255.</td>
</tr>
<tr>
<td>3</td>
<td>Currency</td>
<td>8</td>
<td>Stores money values from $-922,337,203,685,477.5808$ to $+922,337,203,685,477.5807$.</td>
</tr>
<tr>
<td>4</td>
<td>Date</td>
<td>8</td>
<td>Stores date values from 01-Jan-100 to 31-Dec-9999.</td>
</tr>
<tr>
<td>5</td>
<td>Decimal</td>
<td>14</td>
<td>Stores values $\pm 79, 228, 162, 514, 264, 337, 593, 543, 950, 335$ as integer or $\pm 7.2998162514264337593543950335$ as floating point number (28 decimal places).</td>
</tr>
<tr>
<td>6</td>
<td>Double</td>
<td>8</td>
<td>Stores values $\pm 1.79769313486232E308$ to $\pm 4.94065645841247E324$.</td>
</tr>
<tr>
<td>7</td>
<td>Integer</td>
<td>2</td>
<td>Stores values $-32,768$ to $+2,767$.</td>
</tr>
<tr>
<td>8</td>
<td>Long</td>
<td>4</td>
<td>Stores values $-2,147,483,648$ to $+2,147,483,647$.</td>
</tr>
<tr>
<td>9</td>
<td>Object</td>
<td>4</td>
<td>Stores reference to any object.</td>
</tr>
<tr>
<td>10</td>
<td>Single</td>
<td>4</td>
<td>Stores values $\pm 3402823E38$ to $\pm 1.401298E45$.</td>
</tr>
<tr>
<td>11</td>
<td>String</td>
<td>10+length</td>
<td>Stores character values from 1 character to 65,400 characters.</td>
</tr>
<tr>
<td>12</td>
<td>Variant</td>
<td>16 or 22+len</td>
<td>Stores numbers larger than Double or character sequences longer than String types.</td>
</tr>
</tbody>
</table>
Rules for variable names in VBA

1. All variable names should begin with an alphabet.
2. VBA is not case sensitive. A variable declared
   \emph{DIM x as Double}
   can be accessed using a capital \textit{X} as follows:
   \textit{X=50}
   \textit{MsgBox X}.
3. A variable in VBA can have maximum length of 255 characters.
4. The first letter of variable name may be followed by numbers.
5. Special characters like ’ ’ (space), . (dot), ! (exclamation), @, &,$ and \# should not be used in variable names.
6. Variable names should not be the same as function names, method names or predefined constant names.
VBA Shortcuts

1. **Alt+F11**: Opens VBA editor.
2. **Alt+F8**: Displays list of macros.
3. **Ctrl+Break**: Stop execution of program.
4. **Ctrl+G**: Displays immediate window.
5. **Ctrl+I**: Display information regarding selected element.
6. **Ctrl+J**: Displays properties and methods of selected object.
7. **Ctrl+R**: Displays project explorer.
8. **Ctrl+Shift+I**: Display parameter information for selected element.
9. **Ctrl+Shift+J**: List all constants.
10. **F9**: Add breakpoint to examine values in variables.
11. **F5**: Run the program.
12. **F1/ F2/ F4**: Displays Help/ Object browser/ Properties windows.
Useful function in VBA

1 **Asc/ AscB/ AscW**: Convert a character to numeric value in American Standard Code for Information Interchange (ASCII), Double Byte Character Set (DBCS) or Unicode system.

2 **Chr/ ChrB/ ChrW**: Convert a number to character value value in American Standard Code for Information Interchange (ASCII), Double Byte Character Set (DBCS) or Unicode system.

3 **CStr**: Convert any data to String.

4 **CVErr**: Create custom error code for program.

5 **Format**: Converts given date or number value to specified string format.

6 **InputBox**: Displays a dialog and takes single input from user.

7 **MsgBox**: Displays a message box.

8 **Val, CByte, CCur, CDbI, CDec, CInt, CLng, or CSng**: Convert given string to Number, Byte, Currency, Double, Decimal, Integer, Long or Single values.
Operators in VBA

1 VBA Supports the following types of operators:

(i) Arithmetic operators (Table 8)
(ii) Comparison operators (Table 9)
(iii) Concatenation operators (Table 10)
(iv) Logical operators (Table 12)
(v) Bitwise operators (Table ??)
Arithmetic operators in VBA

Arithmetic operators in VBA are enlisted in Table 8.

Table: Arithmetic operators in VBA

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Operator</th>
<th>Purpose</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>^</td>
<td>Power (e.g. $x^y, 2^3$)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>%</td>
<td>Modulo remainder (e.g. $x%y, 22%3$)</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>/</td>
<td>Division (e.g. $x/y, 22/3$)</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>*</td>
<td>Division (e.g. $x<em>y, 22</em>3$)</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>Division (e.g. $x - y, 22 - 3$)</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>+</td>
<td>Division (e.g. $x + y, 22 + 3$)</td>
<td>6</td>
</tr>
</tbody>
</table>
Comparison operators in VBA

Comparison operators in VBA are enlisted in Table 9.

Table: Comparison operators in VBA

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Operator</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>=</td>
<td>Compares for equality (e.g. if $x = y$, if 2 = 3)</td>
</tr>
<tr>
<td>2</td>
<td>&lt;&gt;</td>
<td>Not equal to (e.g. if $x &lt;&gt; y$, if 22 &lt;&gt; 3)</td>
</tr>
<tr>
<td>3</td>
<td>&gt;</td>
<td>Greater than (e.g. if $x &gt; y$, if 22 &gt; 3)</td>
</tr>
<tr>
<td>4</td>
<td>&lt;</td>
<td>Less than (e.g. if $x &lt; y$, if 22 &lt; 3)</td>
</tr>
<tr>
<td>5</td>
<td>&gt;=</td>
<td>Less than or equal to (e.g. if $x &gt;= y$, if 22 &gt;= 3)</td>
</tr>
<tr>
<td>6</td>
<td>&lt;=</td>
<td>Less than or equal to (e.g. if $x &lt;= y$, if 22 &lt;= 3)</td>
</tr>
</tbody>
</table>
Concatenation operators in VBA

Concatenation operators in VBA are enlisted in Table 10.

Table: Concatenation operators in VBA

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Operator</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
<td>Concatenates strings, but adds numbers. (e.g. ( a = &quot;Welcome&quot; + &quot;toVBA!&quot;; b = &quot;5&quot; + &quot;10&quot; ) store &quot;Welcome to VBA!&quot; in ( a ) and 15 in ( b ).)</td>
</tr>
<tr>
<td>2</td>
<td>&amp;</td>
<td>Concatenates strings and numbers. (e.g. ( a = &quot;Welcome&quot; &amp; &quot;toVBA!&quot;; b = &quot;5&quot; &amp; &quot;10&quot; ) store &quot;Welcome to VBA!&quot; in ( a ) and 510 in ( b ).)</td>
</tr>
</tbody>
</table>
Logical operators in VBA

Logical operators in VBA are enlisted in Table 12.

Table: Logical operators in VBA

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Operator</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AND</td>
<td>Returns true if both conditions are true (e.g. If 5 &lt; 10 AND 6 &gt; 10 Then results in false.)</td>
</tr>
<tr>
<td>2</td>
<td>OR</td>
<td>Returns true if at least one condition is true (e.g. If 5 &lt; 10 OR 6 &gt; 10 Then results in true.)</td>
</tr>
<tr>
<td>3</td>
<td>NOT</td>
<td>Negates the result (toggles true or false) (e.g. If NOT 5 &lt; 10 Then results in false.)</td>
</tr>
<tr>
<td>4</td>
<td>XOR</td>
<td>Returns true if and only if only one of the 2 conditions is true (e.g. If 5 &lt; 10 XOR 6 &gt; 10 Then results in true but If 5 &lt; 10 XOR 6 &lt; 10 Then results in false.)</td>
</tr>
</tbody>
</table>
Bitwise operators in VBA

Bitwise operators in VBA are enlisted in Table ??.

### Table: Bitwise operators in VBA

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>Operator</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AND</td>
<td>Returns 1 if both bits are 1 (e.g. ( a = 5 \ AND 3 ) (00000101 ( AND ) 00000011 = 00000001) results in 1.)</td>
</tr>
<tr>
<td>2</td>
<td>OR</td>
<td>Returns 1 if any one of the bits is 1 (e.g. ( a = 5 \ OR 3 ) (00000101 ( OR ) 00000011 = 00000111) results in 7.)</td>
</tr>
<tr>
<td>3</td>
<td>NOT</td>
<td>Returns 1 if the input is 0. Returns 0 if the input is 1. (e.g. ( a = NOT 5 ) (NOT 00000101 = 11111010) results in 250 in byte range.)</td>
</tr>
<tr>
<td>4</td>
<td>XOR</td>
<td>Returns 1 if only one of the given bits is 1. (e.g. ( a = 5 \ AND 3 ) (00000101 ( XOR ) 00000011 = 00000110) results in 6.)</td>
</tr>
</tbody>
</table>
Mathematical Functions in VBA

(1) **ABS**: Returns positive (absolute) value irrespective of whether the input value is positive or negative. e.g. \( ABS(-7) \) returns 7).

(2) **ATN**: Arc tangent or \( \tan^{-1} \)

(3) **COS**: Cosine value.

(4) **EXP**: Exponential value of given number (\( e^x \)).

(5) **FIX**: Return integer portion of given number \( (a = FIX(5.95) \) results in a value of 5).

(6) **FORMAT**: Return the given number formatted as the instructions provided through a string \( (FORMAT(15.95345, "###, 000.00") \) returns 015.95). The value of 0 denotes a compulsory digit and # denotes an optional digit.

(7) **INT**: Return integer portion of given number \( (a = FIX(5.95) \) results in a value of 5).
Mathematical Functions in VBA

(8) **LOG**: Returns logarithm of given number to specified base ($x = \log(2, 10)$) calculates logarithm of 2 to the base 10 and stores the result in $x$.

(9) **RANDOMIZE**: Initializes random number generator to a new initial value.

(10) **RND**: Returns a random integer value between specified input values.

(11) **ROUND**: Returns given number after rounding it to specified digits.

(12) **SGN**: Returns the sign ($\pm$) of given number.

(13) **SIN**: Returns the sine value of given angle.

(14) **SQR**: Returns the square root of given number.

(15) **TAN**: Returns the tangent value of given angle.
String functions in VBA

(1) **ASC**: Returns ASCII (American Standard Code for Information Interchange) number value for given character.

(2) **CHR**: Returns character value of given number based on ASCII (American Standard Code for Information Interchange) system.

(3) **&**: Concatenate 2 strings.

(4) **FORMAT**: Format given string based on the instruction given in the second string. Typical instructions include "Standard", "Percent", "Currency", "Short date", etc.

(5) **INSTR**: Returns the location value at which the second string occurs in the first string.

(6) **INSTRREV**: Returns the location value at which the second string occurs in the first string when searching from the end (reverse direction) of string.

(7) **LCASE**: Returns Lower Case version of the string.

(8) **LEFT**: Returns given number of characters from the left end of string.
String functions in VBA

(9) **RIGHT**: Returns given number of characters from the right end of string.

(10) **LEN**: Returns the length of string.

(11) **LTRIM**: Returns a string after removing all white spaces to the left of the string (leading white spaces).

(12) **RTRIM**: Returns a string after removing all white spaces to the right of the string (trailing white spaces).

(13) **MID**: Extract a substring from given start position through given number of characters.

(14) **REPLACE**: Replace given search string (string 2) using the replacement string (string 3) in the original string (string 1).

(14) **SPACE**: Returns a string containing specified number of spaces.
String functions in VBA

(15) **STR**: Converts a number to String.

(16) **STRCOMP**: Compare 2 strings and return 0 for equal, -1 for less than and 2 for greater than results.

(16) **STRCONV**: Converts given string to upper case, lower case, proper case or unicode as the per the second argument.

(17) **STRREVERSE**: Reverses given string.

(18) **STRREVERSE**: Reverses given string.

(19) **TRIM**: Remove leading and trailing spaces.

(20) **UCASE**: Converts given string to upper case.

(21) **VAL**: Returns numeric value parsed from given string.
Date/ time functions in VBA

1. **DATE**: Returns system date.
2. **DATEADD**: Adds given duration of time (yyyy - year, q - quarter, m - month, y - day of the year, d - day, w - weekday, ww - week, h - hour, n - minute, s - second).
3. **DATEDIFF**: Calculate the difference between date1 and date2 in the unit provided as the first argument (yyyy - year, q - quarter, m - month, y - day of the year, d - day, w - weekday, ww - week, h - hour, n - minute, s - second).
4. **DATEPART**: Return specified part (yyyy - year, m - month, d - day of month) of date object.
5. **DATESERIAL**: Convert given year, month and date values to a date object.
6. **DATEVALUE**: Parses a string containing date value and returns date object.
7. **DAY**: Returns day of month for given date object.
8. **FORMAT(DT, ”FMT”)**: Returns given date formatted as per the instructions provided through the second argument.
Date/ time functions in VBA

(9) **HOUR**: Returns hour part from given time object.
(10) **MINUTE**: Returns minute value of given time object.
(11) **MONTH**: Returns month part of given date object.
(12) **MONTHNAME**: Returns string value of month represented by date object.
(13) **NOW**: Returns current date and time.
(14) **TIMESERIAL**: Returns time object, given the hour, minute and second values.
(14) **TIMEVALUE**: Parses time from given string.
(15) **WEEKDAY**: Returns the number representing the day of week.
(16) **WEEKDAY**: Returns day of week for given date value.
(17) **WEEKDAYNAME**: Returns the name of week day.
(18) **YEAR**: Returns year part of given date object.
Arrays in VBA

1. Arrays in VBA are indexed from 0 to given number. Hence an array declared `Dim x(4) as Variant` contains 5 elements numbered from 0 to 4.

2. Arrays can be declared without specifying size. `Dim a()` is a valid declaration. But, use `reDim` to change its size before using, as in `reDim Preserve a(10)`. The preserve keyword saves any data stored in the previous array. If preserve keyword is omitted (as in `reDim a(10)`), the new array will not have previous data.

3. An array can be declared with initialization data as follows:
   
   ```vba
   Dim a
   a = Array("Coffee","Tea","Juice")
   ```

4. Two dimensional arrays can be declared as follows:
   
   ```vba
   Dim a(5,5) declares a 2 dimensional array having 6 rows and 6 columns indexed from 0 to 5.
   ```
Arrays in VBA

5 Array elements can be populated using the index values as shown below:
   Dim a(3), b(4,4)
   a(0)="Apple"
   a(1)="Grape"
   b(0,0)=5
   b(0,1)=4
   b(1,2)=7

6 Values stored in an array can be accessed using the index values, as shown below:
   Dim a(4) as Variant
   a(0)=5
   a(1)=3
   a(2)=4*a(0)
   msgbox("Result is " & 5*a(1))
Methods of Array in VBA

1 **LBOUND**: Returns the smallest index or lower bound of given array.

2 **UBOUND**: Returns the highest index or upper bound of given array.

3 **SPLIT**: Split given string using a delimiter string and return the array. An example of split:
   Dim a as Variant
   a = Split(”Apple#Orange#Grape”,”#”)
   MsgBox a(1)
displays Orange in the message box.

4 **JOIN**: Join members of an array, using the second parameter as delimiter and return a single string. An example of join:
   Dim a, b as Variant
   a = Array(”Apple”,’”Orange”’,’”Grape”’,’”Strawberry”’)
   b = JOIN(a,”#”)
   MsgBox b
displays ”Apple#Orange#Grape#Strawberry” in the message box.
Methods of Array in VBA

5 **FILTER**: Returns an elements of an array that fulfill given search criterion. Example of filter:

```vba
Dim a, b as Variant
a=Array(“Apple”,”Orange”,”Grape”)
b = FILTER(a,”ra”)
MsgBox JOIN(b,”, ”)
displays Orange, Grape in the message box, since these two elements contain the letters ra in their text.
```

6 **ISARRAY**: Returns true if the given variable is an array. Otherwise, it returns false.

7 **ERASE**: Sets all members of a static array to 0 or ””. Frees memory used by dynamic array (created using REDIM).
Conditions in VBA

Following are the conditions available in VBA for decision making:

1. If ... End If statement
2. If ... Else ... End If statement
3. If ... ElseIf ... End If statement
4. Select ... Case ... End Select statement

A detailed description of each decision making statement is presented in the following subsections.
If ... End If condition

1  If condition takes a comparison and executes the following lines if the condition is true. It may optionally have an Else part which is executed in case the condition is false.

2  The following shows use of if condition:
   Dim x as Integer
   x = InputBox(“Enter your age”, “Age please”)
   If x < 12 Then
   MsgBox “You get an Ice cream!”
   End If
If ... Else ... End If condition

1 The lines following If are executed in case the condition is true. The lines following Else part are executed in case the condition is false.

2 The following shows use of if ... else condition:
   Dim x as Integer
   x = CInt(InputBox(“Enter your age”, “Age please”))
   If x < 12 Then
      MsgBox “You get an Ice cream!”
   Else
      MsgBox “You get a chocolate!”
   End If
If ... Elselse ... End If condition

1 The lines following If are executed in case the condition is true. The Elselse condition is checked when the first condition fails. The ladder moves down till an End If is reached.

2 The following shows use of if ... else condition:

```vba
Dim x as Integer
x = CInt(InputBox(“Enter your age”, “Age please”))
If x < 20 Then
    MsgBox “You get an Ice cream!”
ElseIf x < 20
    MsgBox “You get a Chocolate!”
ElseIf x < 30
    MsgBox “You get a Juice!”
Else
    MsgBox “Sorry! You get nothing!!”
End If
```
Select Case ... End Select condition

1 Select Case statement takes a single variable and compares its value against several case conditions. When no case matches the given select, Case Else (if provided) is executed.

2 Select Case statement ends with End Select.

3 Following is an example of Select Case condition:
   
   ```vba
   Dim c, x, y, z as Integer
   x = CInt(InputBox(“Enter x:”, “Enter number”))
   y = CInt(InputBox(“Enter y:”, “Enter number”))
   c = CInt(InputBox(“Make a choice” & Chr(10) & “1 - Add” & Chr(10) & “2 - Subtract” & Chr(10) & “3 - Multiply” & Chr(10) & “4 - Divide” , “Enter choice”))
   Select Case c
   Case 1
   z = x+y
   MsgBox z
   Case 2
   z = x - y
   MsgBox z
   Case 3
   z = x * y
   MsgBox z
   Case 4
   z = x / y
   MsgBox z
   Case Else
   MsgBox(“Invalid choice”)
   End Select
   ```
Loops in VBA

Following are the types of loop supported by VBA:

1. For loop
2. For ... each loop
3. While ... Wend loop
4. Do While ... Loop
5. Do Until ... Loop

The following sections describe the use of loops listed above.
For loop

1. For loop has start value, end value and a step increment. The for loop ends with Next statement.

2. The following code depicts the use of For loop to calculate the sum of first n integers:

   Dim i, n, sum as Integer
   sum = 0
   n = CInt(InputBox(“Enter n:”, “Enter limit for summation”))
   For i = 1 To n Step 1
     sum = sum + i
   Next
   MsgBox(“1+2+… +” & n & “ = ” & sum)
For Each loop

1. For Each loop is applicable for Arrays. Each element of an array is iterated using the loop and is assigned in a variable. The For End loop ends with Next statement.

2. The following code depicts the use of For End loop to calculate the sum of number elements in an array:

```vba
Dim x, sum as Variant
sum = 0
x = Array(2,6,3,5,1,6)
For Each y in x
    sum = sum + y
Next
MsgBox("Sum = " & sum)
```
While ... Wend loop

1. While loop takes a condition and executes till that condition is satisfied. While loop ends with Wend.

2. The following code depicts the use of While ... Wend loop to calculate the factorial of given number:

   ```vba
   Dim i, n, fact as Integer
   fact = 1
   i = 1
   n = CInt(InputBox("Enter n:“, “Factorial input“))
   While i <= n
     fact = fact * i
     i = i + 1
   Wend
   MsgBox(n & “! = ” & fact)
   ```
Do While ... Loop

1. Do While loop takes a condition and executes till that condition is satisfied. Do While loop ends with Loop statement.

2. The following code depicts the use of Do While ... Loop to display Fibonacci series:
   
   ```vba
   Dim a, b, c, n, i as Integer
   a = b = i = 1
   n = CInt(InputBox("Enter number of terms to display in Fibonacci series:","Number of terms"))
   Do While i <= n
   If i <= 2 Then
       c = 1
   Else
       c = a + b
   a = b
   b = c
   End If
   MsgBox("Term " & i & " of Fibo = " & c)
   i = i + 1
   Loop
   ```
Do Until ... Loop

1. Do Until loop takes a condition and executes till that condition is false. Once the condition becomes true, the loop exits. Do Until loop ends with Loop statement.

2. The following code depicts the use of Do Until ... Loop to display Fibonacci series:

```vbnet
Dim a, b, c, n, i as Integer
a = b = i = 1
n = CInt(InputBox("Enter number of terms to display in Fibonacci series:","Number of terms"))
Do Until i>n
  If i <= 2 Then
    c=1
  Else
    c = a+b
    a=b
    b=c
  End If
  MsgBox("Term " & i & " of Fibo = " & c)
i = i + 1
Loop
```
Message Box in VBA

1 Message Box is displayed using MsgBox function in VBA. MsgBox function takes one String value as a mandatory parameter.

2 The second parameter of message box is a list of numbers or predefined numerical constants to customize the message box.
3 The integer parameters to control VBA dialog are given as follows:

1. **0 or vbOKOnly** - Displays OK button only.
2. **1 or vbOKCancel** - Displays OK and Cancel buttons.
3. **2 or vbAbortRetryIgnore** - Displays Abort, Retry, and Ignore buttons.
4. **3 or vbYesNoCancel** - Displays Yes, No, and Cancel buttons.
5. **4 or vbYesNo** - Displays Yes and No buttons.
6. **5 or vbRetryCancel** - Displays Retry and Cancel buttons.
7. **16 or vbCritical** - Displays Critical Message icon.
8. **32 or vbQuestion** - Displays Warning Query icon.
9. **48 or vbExclamation** - Displays Warning Message icon.
10. **64 or vbInformation** - Displays Information Message icon.
11. **0 or vbDefaultButton1** - First button is default.
12. **256 or vbDefaultButton2** - Second button is default.
13. **512 or vbDefaultButton3** - Third button is default.
14. **768 or vbDefaultButton4** - Fourth button is default.
15. **0 or vbApplicationModal** - The current application will not work until the user responds to the message box.
16. **4096 or vbSystemModal** - The application will keep working, irrespective of whether the user responds to the dialog or not.
3  The integer parameters to returned by VBA dialog based on the response of the user are enlisted as follows:

1. 1 or vbOK - OK was clicked.
2. 2 or vbCancel - Cancel was clicked.
3. 3 or vbAbort - Abort was clicked.
4. 4 or vbRetry - Retry was clicked.
5. 5 or vblgnore - Ignore was clicked.
6. 6 or vbYes - Yes was clicked.
7. 7 or vbNo - No was clicked.
Input Box in VBA

1. Input Box can be displayed using InputBox function in VBA. This function takes one mandatory parameter, which is displayed as the prompt message for input.

2. The second parameter for InputBox is the title tile of the dialog.

3. The third parameter is displayed as the default/suggested input value in the input dialog.

4. The fourth parameter takes x-axis value and the fifth parameter takes y-axis value for displaying the input dialog.

5. The sixth parameter takes the name of help file and the seventh parameter takes help context. It is provided only if the sixth parameter is a help file.
Creating function in VBA

1. Function can be created by double clicking on any component and entering the coding window. Delete the default subroutine in which the cursor lands.

2. Function should begin with the keyword function and is followed by name of the function.

3. Name of the function is followed by opening and closing parentheses.

4. The parameters to be passed to the function are enlisted inside the parentheses.

5. The result or return value of the function should be assigned to a variable having the same name as the function.

6. The function can be invoked from the spreadsheet as if it were a predefined Excel function.

7. Function ends with End Function keyword.
Example of user defined function in VBA

The following function takes single input parameter and prints it 10 times using a for loop:

```vba
Function rep(x as String)
Dim r, i as Variant
r = ""
For i = 1 To 10 Step 1
r = r & x
Next
rep = r
End Function
```
Creating and editing procedures/subroutines in VBA

1. Subroutine begins with the keyword `sub` and is followed by the name of the subroutine.
2. Subroutine ends with the keyword `End Sub`.
3. Subroutine does not return any value. It may store result in global arrays.
4. Subroutines are used to handle events related to user interface components.
Example of subroutine or procedure in VBA

The following subroutine is invoked by clicking CommandButton1 placed on the userform:

```vba
Sub CommandButton1_Clicked()
    Dim l, b, a as Double
    l = CDbl(InputBox("Enter length"))
    b = CDbl(InputBox("Enter width"))
    a = l * b
    MsgBox("Area = " & a)
End Sub
```
Creating and editing macros in VBA

1. Macro can be recorded from Macro icon found in View menu.
2. It opens a dialog to record new macro.
3. A button or keyboard shortcut may be assigned to record the macro.
4. Excel automatically records the commands equivalent to the actions performed.
5. Macro may be edited by choosing the macro name and pressing the Edit button in the macro dialog.
Concept of Object Oriented Programming

1. VBA supports creation of classes (using Type keyword), properties, functions and subroutines. VBA does not support inheritance.

2. A new class can be created using Insert → Class Module. Suitably name the class module.

3. The class module may contain variables for storing data related to the class.

4. Property may be declared with access level (Public or Private) and access type defined by GET, LET/SET.

5. Subroutines and functions created inside the class module are called methods.

6. VBA creates 2 automatic subroutines named Private Sub Class_Initialize() and Private Sub Class_Terminate(), which are invoked at the time of loading the objects of class and clearing the object from memory respectively.

7. Properties and methods of a class may be accessed using dot (.) operator.

8. Objects are instantiated using New keyword.
Creating class in VBA

1. Open MS Excel. Press Alt+F11 to open VBA IDE.
2. Choose Insert → Class Module (Alt+I+C). A class module called Class1 is displayed. Change its name to Student.
3. Type the following content in the class module to create a Student class having properties Name, DOB, Mark and Total. The class has methods named age and percentage. The class defines variables named n, d, m, t to hold values of name, date of birth, mark and total mark respectively.
Private n as String
Private d as Date
Private m, t as Double
Public Property Let Name(x as String)
n = x
End Property
Public Property Get Name() as String
name = n
End Property
Public Property Let DOB(x as Date)
d = x
End Property
Public Property Get DOB() as Date
DOB = d
End Property
Public Property Let Mark(x as Double)
m = x
End Property
Public Property Get Mark() as Double
Mark = m
End Property
Public Property Let Total(x as Double)
t = x
End Property
Public Property Get Total() as Double
Total = t
End Property
Public Function age() as String
age = DateDiff(“YYYY”,d, Date)
End Function
Public Function Percentage() as Double
Percentage = m / t * 100
End Function
Public Sub howAreYou()
MsgBox (“Fine!”)
End Sub
4 Insert userform by choosing `Insert → User Form (Alt+I+U)`.

5 Insert 4 labels and modify their captions to “Name, DOB, Mark and Total” respectively.

6 Insert 4 Textboxes (named `TextBox1` to `TextBox4`) to get 4 input values.

7 Insert a command button (`CommandButton1`). Change its caption to `Test OOPS`. 
8 Enter the following code to handle clicking of CommandButton1:

```vbnet
Private Sub CommandButton1_Click()
    Dim x as New Student
    x.Name = TextBox1.Text
    x.DOB = CDate(TextBox2.Text)
    x.Mark = CDbl(TextBox3.Text)
    x.Total = CDbl(TextBox4.Text)
    MsgBox (x.Name & Chr(10) & x.DOB & Chr(10) & x.Mark & x.Total)
    MsgBox ("Age = " & x.age & " years" & Chr(10) & "Percentage = " & x.percentage & "%")
    x.howAreYou
End Sub
```

9 Run the program by pressing F5 and check whether the program works correctly.
Concept of event driven programming

1. In the normal programming, prompts are displayed to collect input and manage user interaction.

2. Event driven programming displays a passive user interface, which starts working when the user performs certain actions, like clicking a button, typing, moving the mouse, etc.

3. Event driven programming is the king pin for Graphical User Interface (GUI), since components fire events.

4. Events fired by components may be handled by designated subroutines or functions.

5. In VBA, the most common prompt component is a label, input component is a TextBox and action component is a CommandButton.
6 In the coding window, VBA permits selection of component (top left of the coding window) and an associated event (top right of the coding window) which may be handled by a subroutine.

7 Event driven programming has made it possible to use menu, combobox, list, radiobutton, checkbox, button and other components based on user interaction.

8 Components awaiting user interaction are similar to servants awaiting command from the employer. Once an interaction occurs, a particular job is carried out using event handlers.
1 **UserForm**: UserForm is the control used for display of a window.

2 **Label**: Label displays text. Caption property is used to set or get text.

3 **TextBox**: TextBox takes text input. The Text property is used to set or get the value.

4 **ComboBox**: Displays a ComboBox control which displays several options to select from. It is initialized using UserForm_Activate() method using AddItem property of the combobox control.

5 **ListBox**: ListBox displays a scrollable list of items. The elements of ListBox may be added by handling UserForm_Activate() method and selecting AddItem property.
Forms and controls in VBA

6 **CheckBox**: CheckBox provides a tickable control. Display value of CheckBox can be controlled using Caption property. Tick status can be set or unset using Value property. When Value property is true, tick mark appears. When Value property is false, tick mark disappears.

7 **CheckBox**: CheckBox provides a tickable control. Display value of CheckBox can be controlled using Caption property. Tick status can be set or unset using Value property. When Value property is true, tick mark appears. When Value property is false, tick mark disappears.

8 **OptionBox**: OptionBox provides the facility to select one option from several options. Normally, group of OptionBoxes are placed on a Frame control to enable group behaviour. OptionBoxes placed on same parent control (like Frame, UserForm, etc.) behave as a group (only one gets selected from a group). Value property of OptionBox determines whether an OptionBox is selected or not.
9 **ToggleButton**: ToggleButton keeps in pressed condition (Value=true) when clicked once. The button gets released when clicked again (Value=false). The pressed or released condition is controlled based on True or False stored in Value property.

10 **TabStrip**: TabStrip displays Tab1, Tab2, etc. and permits selection of different set of components for each tab. Component behaviour can be controlled using TabStrip1_Change() subroutine.

10 **TabStrip**: TabStrip displays Tab1, Tab2, etc. and permits selection of different set of components for each tab. Component behaviour can be controlled using TabStrip1_Change() subroutine.

11 **MultiPage**: MultiPage displays Page1, Page2, etc. and permits selection of different set of components for each page. Component behaviour can be controlled using MultiPage1_Change() subroutine.
Forms and controls in VBA

12 **MultiPage**: MultiPage displays Page1, Page2, etc. and permits selection of different set of components for each page. Component behaviour can be controlled using MultiPage1_Change() subroutine.

13 **SpinBox**: SpinBox can be inserted in a form near any numeric input control (like TextBox). On SpinBox1_Change() method can be used to get the value from the SpinBox and set the same on TextBox or other control.

14 **Image**: Image control displays any image provided through Picture property.

15 **RefEdit**: RefEdit helps to select a range of cells from spreadsheet during the execution of the program. Selected range is stored in Value property of the RefEdit object.
Properties, events and methods associated with VBA controls

1. Properties are the values which can control the behaviour of an object.

2. The most commonly used properties are Caption, Text, Value, Font, etc.

3. Properties window can be made visible or invisible by choosing View→Properties Window or by pressing Alt+V+W or by pressing F4.

4. All the properties enlisted in the properties window can be accessed by placing a dot (.) after the name of the object.
Events associated with VBA controls

1. Events get activated when particular action occurs due to user interaction.
2. Events associated with a particular control can be chosen from the right side ComboBox of coding window.
3. Following are the most common events:
   (i) Activate
   (ii) BeforeDragOver
   (iii) BeforeDropOrPaste
   (iv) Click
   (v) DblClick
   (vi) Deactivate
   (vii) Error
   (viii) Initialize
   (ix) KeyDown
   (x) KeyPress
   (xi) KeyUp
   (xii) MouseDown
   (xiii) MouseMove
   (xiv) MouseUp
   (xv) QueryClose - Only for UserForm
Methods associated with VBA controls

1. Subroutines handle events triggered by the controls.
2. The methods or subroutines have two parts, separated by an underscore (_).
Using ActiveX data objects in VBA

1. VBA allows database interaction using ADO objects.

2. Several types of databases are supported. The connection string applicable for particular database type is to be chosen.

3. VBA support for particular type of database needs to be enabled from Tools → References. Select Microsoft ADODC Connection Object having appropriate version number.

4. An ADODC object needs to be initialized as follows:
   
   ```vba
   Dim conn as New ADODB.Connection
   ```

5. A recordset object needs to be initialized as follows:
   
   ```vba
   Dim rs as New ADODB.Recordset
   ```
Using ActiveX data objects in VBA

6 A recordset object needs to be initialized as follows:
   Dim rs as New ADODB.Recordset

7 Open method of ADODB.Connection object is to be invoked by providing a connection string suitable for the selected database engine and the data source.

8 Select query may be executed by replacing table name with sheet name as follows:
   select * from [Sheet1$];

9 Range object of Excel allows Copying of result set values in selected range of cells.
Debugging techniques

1. The IDE provided with VBA points out simple mistakes committed during coding (like unclosed brackets, forgotten double quotes, etc.)

2. Certain errors occur during the runtime. When error window appears, click Debug instead of End to start debugging.

3. Runtime errors are identified by selecting the line containing the error.

4. Just hover the mouse over the variables in the error line. The present value of the variable would be displayed. If the value is not the value expected to be stored in the variable, trace the line in which assignment to that variable takes place.

5. Leaving an object without a proper initial value may cause error. Please ensure that the objects in the error line have already been initialized.

6. Open Immediate Window (View -> Immediate Window or press Ctrl+G). Enter any test code in the immediate window and check the results.

6. Create a breakpoint by Clicking on the margin to the right of the line requiring a break. You could also press F9 to set breakpoint.
Debugging techniques

7 The code stops executing at breakpoint. The values store in variables may be checked from the coding window.

8 To move execution to a particular line, choose Debug— >Step Into or press F8.

9 To move execution past the current line, choose Debug— >Step Over or Press Shift+F8.

10 You can execute the code one line at a time by repeatedly choosing Debug— >Step Over or Pressing Shift+F8.

10 You can halt the line by line execution of break point using Debug— >Step Out (Ctrl+Shift+F8).

11 You can choose to continue execution using Run— >Continue from the menu.

12 You could break execution of program by choosing Run— >Break from the menu or by pressing Ctrl+Break.

13 The entire execution can be reset by pressing Run— >Reset.
Smart accounting

Basics of Accounting

1. Accounting the recording, verification and tallying of monetary transactions.
2. Accounting transaction may relate to assets, liabilities, income, expenditure, inventory, etc.
3. Accounting transaction are chronological in nature, which means each transaction contains a date.
4. Accounts are recorded in 8 column format.
5. First 4 columns of the account sheet are meant for Debit (Dr.).
6. Columns 5 to 8 of the account sheet are meant for Credit (Cr.).
Basics of Accounting

7 Column 1 is meant for date on the debit side. Column 5 is meant for date on the credit side.

8 Column 2 is meant for particulars on debit side. Column 6 is meant for particulars on the credit side.

9 Column 3 is meant for Journal Folio on debit side. Column 7 is meant for Journal Folio on the credit side.

10 Column 4 is meant for amount on debit side. Column 8 is meant for amount on the credit side.

11 This method of recoding accounts in 8 columns of debit and credit is called **T-Accounting**.

12 Accounting has 3 major components:
   (i) Recording
   (ii) Classifying
   (iii) Summarising
1 **American method**: American method is based on the following accounts:

(i) **Assets account**: Considers all items which can be disposed and converted to money.

(ii) **Liabilities account**: Considers all items for which money should be earmarked.

(iii) **Revenue account**: Considers all transactions which will increase the monetary strength.

(iv) **Expense account**: Considers all transactions which result in reduction of monetary strength.

2 **British method**: British method is based on the following accounts:

(i) **Personal accounts**: Accounts which are attributed to specific individual or business source. All balances are classified under personal accounts.

(ii) **Real accounts**: Accounts related to inventories, assets and liabilities.

(iii) **Nominal account**: The account which considers inflow and outflow of money (income, expense, gain, loss, etc.) is called nominal account. Increase in the inflow of money through nominal account makes the business strong.
Accounting concepts

1 **Business entity concept**: The owner and the business are treated as separate entities.

2 **Money measurement concept**: Accounting considers only those transactions which can be measured in terms of money. In other words, non-monetary transactions are not recorded.

3 **Going concern concept**: Accounting assumes that the business as a long lasting process, irrespective of the status of its owner, assets or employees.

4 **Dual aspect concept**: Each credit has a corresponding debit. Credit is recorded in one place and the debit in some other place. But, the sum of debits and the sum of credit should always match.

5 **Cost concept**: Each transaction is recorded as per the cost incurred by the business not based on the market value. In simple terms, accounting records only the actual cost.
Accounting concepts

6 **Accounting period concept**: Although a business is an indefinite process in time (going concern concept), accounts are closed periodically (at the end of a financial year) to determine the profit, loss, balance position, etc. of the business.

7 **Matching cost concept**: For any accounting period, the revenue generated and the cost incurred are matched against each other to determine the net profit or loss for that period.

8 **Accrual concept**: Revenue is recorded on the date of its generation (not the date of actual realization) and expenditure is recorded as the date of incurring the expense (not the date on which money flows out - like the date of drawal of cheque).

9 **Objective evidence concept**: Each transaction entry should be accompanied by corresponding bill, invoice, voucher, receipt, challan, etc. to substantiate the statement and to permit independent verification of the validity of the transaction.
Golden rules of double entry system

1. Each credit should have a corresponding debit and each debit should have a corresponding credit. It means each transaction is recorded in the form a debit and a credit - resulting in double entry system. The double entry system results in the following sub rules:

   (i) **In personal accounts**: Debit the receiver and credit the giver.
   (ii) **In real accounts**: Debit the incoming asset and credit the outgoing asset.
   (iii) **In nominal account**: Debit all expenses and credit all incomes.

2. In general, a transaction is a debit if it decreases money owned by a business and credit if it increases money owned by a business.
Accounting terminologies

1 **Account**: The record of all transactions involving money to ascertain the current status of business is called account.

2 **Accounting**: The process of recording financial transactions is called accounting.

3 **Accrual basis**: Recording the cost when incurred rather than paid and recording revenue when earned rather than collected is called accrual basis (counting the hens before the eggs are hatched). In other words, the date of incurring an expense through cheque may be different from the date on which the amount is drawn by the party and the date of selling a product on gain is different from the date on which the actual revenue reaches the business.

4 **Asset**: Any item owned by the business which can be converted to money is called asset.

5 **Credit**: Credit is the increase in money due to profit, sale of asset or pay-out for a liability. It is posted on the right side of accounting form.
Accounting terminologies

6 **Debit**: Debit is the reduction in money due to loss, purchase of asset or incurring a liability.

7 **Double entry accounting**: Balancing debit at some ledger through a credit at some other ledger is called double entry accounting. Total credit and total debit balance each other.

8 **Expense**: The outflow of money for getting inflow of goods or services (like rent, electricity bill, telephone bill, salary, etc.) is called expense.

9 **Income or revenue**: The inflow of money due to outflow of products or services is called income.
Voucher entry

1 Voucher is a form for entering a financial transaction.
2 Voucher has 2 sides, one for debit and another for credit.
3 A voucher may result in change of assets (inventory voucher) or may result only in change of monetary position (accounting voucher).
4 Voucher is prepared against invoices or other requests for payment. Receipt voucher is prepared against payment received from third party.
5 Vouchers are entered in journal, before posting them to appropriate ledger.
6 The ledger name in which a credit or debit takes place is entered in the journal itself.
Ledger Posting

1. Entering journal entries in ledger of appropriate head of account is called ledger posting.

2. Credit side transaction entries are taken to credit side of the ledger and debit side transaction entries to the debit side of ledger.

3. The ledgers are divided into appropriate heads of account.
Final Accounts Preparation

1. Final accounts are prepared at the end of financial year.
2. Trading account, profit and loss account and balance sheet are some of the final account formats.
3. Final accounts help to ascertain the financial status of business.
Cash Book

2. All cash inflows like bank withdrawal, receipt of sales proceeds in cash, etc. are entered in the debit side of cash book.
3. All cash outflows like salary, rent, phone bill, etc. made using cash are recorded on the credit side of cash book.
Ratio Analysis

1. Ratio is a measure of how good a condition has been achieved.
2. Following are the common ratios adopted in financial management:
   
   (i) **Current ratio** = \( \frac{\text{current assets}}{\text{current liabilities}} \)
   
   (ii) **Quick ratio** = \( \frac{\text{liquid assets}}{\text{current liabilities}} \)
   
   (iii) **Absolute liquid ratio** = \( \frac{\text{absolute liquid assets}}{\text{current liabilities}} \)
   
   (iv) **Gross profit ratio** = \( \frac{\text{gross profit}}{\text{net sales}} \)
   
   (v) **Net profit to Net worth ratio** = \( \frac{\text{net profit after taxes}}{\text{shareholders' net worth}} \times 100 \)
   
   (vi) **Dividend yield ratio** = \( \frac{\text{Dividend per share}}{\text{market value per share}} \times 100 \)
   
   (vii) **Price earnings ratio** = \( \frac{\text{market price per share}}{\text{earning per share}} \times 100 \)
   
   (viii) **Net profit ratio** = \( \frac{\text{operating profit}}{\text{net sales}} \times 100 \)
   
   (ix) **Inventory ratio** = \( \frac{\text{net sales}}{\text{inventory value}} \)
   
   (x) **Overall profitability ratio** = \( \frac{\text{net profit}}{\text{total assets}} \)
Depreciation

- Reduction in value of a product due to aging is called depreciation.
- The rate of depreciation is affected by the wear, tear and damages.
- The value of a product is reduced using certain formulae at the end of every year.
- At the end of every year, the value of asset should be calculated after allowing for cumulative depreciation from the date of purchase to the date in question.

**Asset cost:** The amount of money spent to purchase an asset is called asset cost.

**Life span:** The useful service period of an asset (machinery, vehicle, building, etc.) is called life span.

**Salvage value of an asset:** The monetary value obtained when disposing an asset after its useful life span is called salvage value.
Depreciation formulae

- **Straight line method**: Based on the straight line method, annual rate of depreciation is calculated using a linear formula. This method is suitable for assets having a known lifespan in years.

\[
Book\ value = Asset\ cost - \frac{(Asset\ cost - Salvage\ value)}{Life\ span} \times Age\ in\ years
\]

- **Units of production method**: When the number of units produced during useful lifespan of an asset is known, depreciation till date can be calculated using the following formula:

\[
Book\ value = Asset\ cost - \frac{(Asset\ cost - Salvage\ value)}{Capacity\ of\ machine\ in\ units} \times Number\ of\ units
\]

- **Double depreciation method**: Double depreciation method permits higher depreciation at the beginning of lifespan and lower distribution near the end of life span of an asset.

\[
Book\ value = Asset\ cost - 2 \times Straight\ line\ depreciation \times Book\ value
\]
Stock management

- Purchase, storage and issue of stock is called stock management.
- **Reorder level**: The safe stock level on reaching which new stock may be ordered without affecting production process is called reorder level.
- **Overstocking**: Purchase of excess stock which reduces money flow is called overstocking.
- **Understocking**: Keeping stock levels much lower than the demand levels for maintaining continuous production is called understocking.
- **VAT**: The tax payable for the increase in value of an item due to manufacturing process is called Value Added Tax (VAT).
- VAT is applicable on the difference between sale price of finished product and the cost of raw materials.
- Increase in price of raw materials results in levy of Value Added Tax.
- VAT is applied in several stages in proportion to the amount of increase in value of an item.
- Consider that the cost of raw materials is Rs.200, the price of finished product is Rs.600 and the rate of VAT is 4%. The Value Added Tax payable may be calculated as shown below:

\[
VAT = \left( \text{Cost of finished product} - \text{cost of raw materials} \right) \times \frac{4}{100}
\]

\[
= (600 - 200) \times \frac{4}{100} = 400 \times \frac{4}{100} = \text{Rs.16}
\]
- Several taxes, including VAT, have been replaced by a single tax called Goods and Services Tax (GST).
Cash flow

- The movement of money into and out of a business is called cash flow.
- If the inflow of money is greater than the outflow of money, a business is said to grow.
- If the outflow of money is greater than the inflow of money, a business is said to decline.
- Inflow of money is different from net profit. Inflow of money may be due to profit, loans and proceeds of selling asset.
- Outflow of money is different from net loss to a business. Outflow of money may be due to purchase of inventory, cost of production, purchase of asset, repayment of loan or loss in business.
Fund flow

- Fund is money held in the form cash, account balance and asset.
- Fund flow is inward and outward movement of cash, account balance and asset values of a company.
- Cash flow is one part of fund flow.
- Fund flow is analysed every month and at the end of every quarter (3 months period) of a financial year.
Introduction to Tally

- Tally is an accounting software.
- Tally supports cash accounting, inventory and payroll accounting.
- Tally supports multiple accounts and accounting vouchers.
- Tally can maintain accounts for multiple companies on the same computer.
- Tally supports multiple languages.
- Security for each company can be provided through Tally.
- Tally can maintain tax and liability accounts.
Features and advantages of Tally

▶ Tally has an easy user interface.
▶ Tally allows data entry in many languages.
▶ Tally supports double entry system of accounting where each amount is entered in credit and debit sides to balance the transactions.
▶ Tally allows creation and maintenance of 99,999 companies.
▶ Tally allows generation of several accounting reports.
▶ Tally allows viewing product-wise inventory level.
▶ Tally allows maintaining reorder levels.
▶ It calculates profits, losses, income and expenditure and readily displays the reports.
▶ Tally allows creation of multiple godowns for inventory management.
▶ Tally supports the creation of cost centre and profit centre.
Tally can produce balance sheets and trial balances without much effort.

Tally analyse the all ratios required to ascertain the performance of a company.

It is easy to detect and correct errors in transaction entries.

Since Tally supports double entry system, the account is almost always balanced.

Stock items may have multiple unit types. Tally supports compound units created out of certain basic units.

Tally highlights any negative balance using red letters, so that the result of an overdraft transaction may be viewed even before committing the transaction.
Implementing accounts in Tally

► The first step towards implementing accounts is to create a company. Pressing Al+F3 (Company Info) brings the company menu. Choose Create company to create a new company.

► After creating a company, press F11 to enable features of the company. Enable Payroll, invoicing, budgeting, etc from accounting features. Similarly, inventory features and statutory and taxation features can be enabled from the respective menus under Company Features.

► Configuration of accounting details like Groups, Ledgers, Budget, Scenarios and voucher types is possible using Gateway of Tally— >Accounts Info.

► Payroll may be configured using Gateway of Tally— >Payroll Info. It supports Employee groups, Employees, Units (Work), Attendance/ Production Types, Pay Heads, Salary Details and Voucher Types can be controlled using this menu.

► Inventory details like Stock Groups, Stock Categories (optional), Tariff Classifications, Stock Items, Units of
Double entry system of Book keeping

- Double entry system mandates that each transaction has 2 entries, one under debit and another under credit.
- Under double entry system, an account which takes money is debited and the account which supplies money is credited.
- Double entry system permits balancing of all accounts to check whether the sum of debits is equal to the sum of credits.
- Double entry system relies on the principle that for any debit made to one account, corresponding credit is made to some other account.
- Tally offers budgeting through GoT—> Accounting Info—> Budgets.
Budgeting Systems

- The system of allotting money for specific purposes and ensuring that the purposes are achieved at the end of specific period is called budget.
- Budget can be short term (monthly, quarterly, annual, etc.) or long term (5 year plan, 10 year plan, etc.).
- Budget provides a specific monetary target for achieving desired goals of the year.
- Budgeting provides a control (or capping) towards each item of expenditure.
- Each head of budget can be controlled by separate individual and the works related to a particular head can be accelerated independent of other heads.
- To create a budget item, use GoT → Accounts Info → Budgets → Create.
- Choose GoT → Display → Trial Balance and press Alt+B to display budget details along with the account details.
Scenario management

- Creating several hypothetical vouchers to measure the impact of changes to the accounting system is called scenario management.
- Scenario allows imposes several new changes to be made to the accounts. These changes will help to arrive at a strategy which is suitable for tiding over specific situations.
- Each scenario presents an alternative way out from a difficult situation.
- Creating different scenarios through voucher entries does not affect the original account.
- Scenario is an effort to guess the real world action-reaction sequence to arrive at the right decision.
- Scenario can be created using GoT → Accounts Info → Scenario. Provide a name and include the types of vouchers applicable for scenario.
- Voucher entry for scenario is made using Reversing Journal (F10) under Accounting Vouchers.
- To compare actuals with scenario, open Balance Sheet, create
Variance Analysis

- Variance analysis in Tally aims to compare the difference between actual performance of a company against budget goals.
- Choose GoT ➔ Display ➔ Trial Balance.
- Press Alt+B to display budget heads. Select the type of head to be displayed for comparison with actual expense.
Costing Systems

- Costing is the calculation of the amount of money spent on fulfilling an order or making a product.
- A break up of cost can be provided through the analysis of inventory expenses, salary, overhead expenses, etc.
- Cost centres can be created (using Features (F11) from GoT) to maintain several cost centres for a company.
Concepts of Ratios

▶ Measuring performance of a company against available resources for performance is called ratio.

▶ Ratio is a unitless number, ranging from 0 to 1, which indicates the performance of a company against specific criteria.

▶ Ratio could also be expressed in the form a percentage after multiplying the ratio by 100.

▶ Tally automatically displays the ratios related to a company under GoT— >Ratio Analysis.

▶ The ratios displayed by default on the right side of the ratio analysis screen include current ratio, quick ratio, debt/equity ratio, gross profit, net profit, operating cost, recovery turnover in days, return on investment and return on working capital.
Analysis of financial statements

- Balance sheet & profit and loss are the most important financial statements provided by Tally (as per Schedule VI of the Companies Act).
- Other statements of account are available under GoT → Display → Statement of Accounts. This menu provides Outstanding and Statistics for display of statement.
- A statement of inventory is available under GoT → Display → Statement of Inventory.
Inventory Basics

- Inventory is the stock of items maintained for production purpose.
- Inventory Groups are the basic groups under which Inventory Items can be created.
- Each item may have an optional Category, which should be enabled using Features (F11) from the GoT.
- Units of measure like No., metre etc. can be created using Units of Measure available under GoT – >Inventory Info.
- Two simple units can be combined using appropriate conversion factor to create a compound unit.
- Inventory items can be Purchased (F9) and Sold (F8) through GoT – >Accounting Vouchers.
- Inventory items can be ordered (Purchase Order (Alt+F4)/Sales Order (Alt+F4) GoT – >Accounting Vouchers.)
POS Invoicing

- Tally allows creating Point Of Sale (POS) voucher using GoT— >Accounts Info— >Voucher Types— >Create. Enter POS Voucher against name, Sale against under, Automatic against voucher numbering.

- To create a sales voucher, choose GoT— >Accounting Vouchers, press F8 for sales voucher, select POS Voucher, enter name of buyer, enter delivery note details, select Sales Ledger against sales ledger, Choose items for sale, enter rate. Accept the sales transaction.

- POS transactions are subject to Value Added Tax (VAT). You may configure VAT using F11 key from the Gateway of Tally.
TDS, TCS, FBT, VAT & Service Tax

- **TDS**: Tax Deducted at Source. It is deducted from salary/other payment.
- **TCS**: Tax Collected at Source. It is collected from sale/other receipts.
- **FBT**: Fringe Benefit Tax. It is the tax paid by an employer towards small benefits, coupons, facilities provided to the employees.
- **VAT**: Value Added Tax. Value added tax is levied against goods sold.
- **Service Tax**: The tax levied by the Government against the monetary value of services offered by a firm to its customers.
- In Tally, all taxes can be enabled or disabled using GoT— >Features (F11)— >Statutory & Taxation.
- Tax reports can be printed from GoT— >Display— >Statutory Reports.
Tally Interface in Different Languages

- The interface and reports of tally can be changed to a different language by pressing \texttt{Alt+G} from the Gateway of Tally (GoT).
- Tally ERP 9 supports 13 languages including Tamil. Choose any language of your choice.
Definition of E-Commerce

1. e-commerce stands for electronic commerce, which means selling or buying goods through electronic means like Internet, phone line, sms, etc.

2. Selling or buying goods through electronic means (based on product description and photos) is called E-Commerce.

3. e-commerce may be carried out over phone line, SMS or the Internet.

4. The buyer relies upon the description, photos and videos of the product being sold online to make a decision.

5. e-commerce involves to entities (the buyer and the seller) connected through a common platform.
Types of e-commerce

1. **B2B or Business to Business**: When the seller is a business firm and the buyer too is business firm (not an individual consumer), the e-commerce transaction is called B2B or Business to Business transaction.

2. **B2C or Business to Consumer**: When the seller is an online business firm and the buyer is an individual, the online business transaction is called B2C or Business to Consumer.

3. **C2B or Consumer to Business**: When the seller is an individual and the buyer is a business firm, the online business model is called C2B or Consumer to Business.

4. **C2C or Consumer to Consumer**: When the seller and the buyer are both individuals, the online business model is called C2C or Consumer to Consumer.
Scope of e-commerce platform

1. e-commerce platforms like quikr, olx, etc. have local scope.
2. e-commerce platforms like Flipkart, Shopclues, Snapdeal, etc.
3. e-commerce platforms like eBay, amazon, etc. have global scope in electronic commerce.
4. Scope of the e-commerce platform ranges from local level to global level.
5. Benefits
Benefits of e-commerce

1. Electronic commerce helps the buyer to choose the most competitive price for a product from several alternate sellers.
2. E-commerce allows a seller to fetch competitive price for products.
3. E-commerce helps a consumer to order a product from far off places, even if such product is not available in the local market.
Disadvantages of e-commerce

1. Descriptions and photos provided by the seller may not match the delivered product.
2. A product may get damaged during transit, since the seller and the buyer are located at far off places.
3. The buyer may fail to pay for a product.
4. The buyer might return the product after using it for few days, which could result in the loss of transit and other expenses to the seller.
5. The buyer might get delivery of the product on the expected date.
## Difference between E-Commerce and traditional commerce

<table>
<thead>
<tr>
<th>No.</th>
<th>e-commerce</th>
<th>traditional commerce</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Buyer cannot touch or feel the product before buying.</td>
<td>Buyer can examine the product before buying.</td>
</tr>
<tr>
<td>2</td>
<td>Bargaining is restricted.</td>
<td>Bargaining is possible.</td>
</tr>
<tr>
<td>3</td>
<td>Shipping time results in delay in getting the product.</td>
<td>Immediate delivery of product.</td>
</tr>
<tr>
<td>4</td>
<td>User needs to figure out the usage procedure by himself.</td>
<td>Usually, a demo is given at the shop floor.</td>
</tr>
<tr>
<td>5</td>
<td>Testing a product before purchase is not possible.</td>
<td>Testing before purchase is possible.</td>
</tr>
<tr>
<td>6</td>
<td>In case of dispute, the buyer and seller need to be fluent in a common language (like English).</td>
<td>Conversations regarding a product are easy.</td>
</tr>
<tr>
<td>7</td>
<td>e-commerce platform allows manufacturers to speedily distribute their new products.</td>
<td>Product infiltration through the market chain takes time.</td>
</tr>
<tr>
<td>8</td>
<td>Fraudulant platforms might result in loss to the buyer.</td>
<td>The buyer can be assured of product delivery for the money paid.</td>
</tr>
<tr>
<td>9</td>
<td>e-commerce permits comparison of several products.</td>
<td>Visiting several shops to buy a product is cumbersome.</td>
</tr>
</tbody>
</table>
Capabilities required for e-commerce

1. An e-commerce platform should provide a cart, where items may be added or removed.

2. Options for payment through CoD (Cash on Delivery), credit card, debit card, netbanking and wallet transfer should be supported by the platform.

3. e-commerce platform should provide a dedicated transport mechanism to deliver the products on specified time line.

4. The buyer and seller should be permitted to share and rate their experience for each online transaction.
Technology issues for e-commerce

1. The online platform should provide secure transaction facilities to prevent fraudsters from cheating a buyer or seller.

2. The payment gateway should be secure and be available 24 × 7.

3. Reward points and redemption system should be provided to encourage the buyer.

4. Broken Internet connection before completing a transaction might result in loss of money to the buyer.
Types of E-Commerce web sites

1 **Transactional type sites**: In transactional type, physical stores provide an online alternative. Customers can either visit the physical store or the online store to get a product.

2 **Information delivery sites**: Large manufacturers might run websites simply to disseminate information about their products. These sites might include online fora for discussion on technical issues.

3 **e-market place**: This type of website permits registration of buyers and sellers. The e-marketplace functions as the facilitator between the buyer and the seller and takes commission amount for the service.
Building business on the net

1. Building an online business requires the seller to create visuals and descriptions for the products.
2. The seller should ensure that the description is unambiguous.
3. The seller should keep the buyer updated about each stage of the online transaction from purchase to eventual delivery of product.
4. The seller should ensure that the buyer is provided a pleasant experience through all stages of his transaction.
5. The buyer should have the service and support for the online product.
6. The online business platform should provide an easy interface to the users.
Providing full description of a product sold online is called the online catalogue.

The online catalogue should describe all physical and internal features.

The online catalogue should provide clear instructions for operating a product.

Online catalogue should provide clear price tag for each product.

Online catalogue should describe defects, if any, in the product.
Shopping cart

- The place for keeping all selected items before final checkout from an online store is called a cart.
- Cart provides facility to add, remove or modify products through online.
- Cart items are deleted when a purchase transaction is made online.
Checkout pages

- After adding items to the cart, the customer is provided a detailed list of items, quantities, rates and amounts for each transaction.
- When the user provides confirmation that the items in the cart are really meant to be purchased, the checkout page redirects the user to the payment page.
Payment for e-commerce transaction

- Payment for an online transaction may be made through the following methods:
  
  (i) **Cash On Delivery (COD)**: The buyer pays money when the product is delivered.
  
  (ii) **Credit card**: A card bearing 16 digit number, expiry month & year, name of card holder and a Card Verification Value (CVV) and permitting spending first and collection on later date is called credit card. This card permits payment of money for goods and services. This amount should be paid up during the credit card bill cycle. Credit card permits conversion of large purchase amount to EMI (Equated Monthly Installment) payment.
  
  (iii) **Debit card**: A card bearing 16 digit number, expiry month & year, name of card holder and a Card Verification Value (CVV) is called and permits spending the money already deposited in an account is called debit card. This card permits payment of money for goods and services from prepaid deposits.
  
  (iv) **Internet banking**: Internet banking permits logging in to a bank site using an ID and password and authorizing the online payment. Some banks send an OTP (One Time Password) to
Security issues for online payment

- The user should ensure that green color lock icon appears near the address bar of the browser.
- The user should ensure that the lock icon is not crossed out.
- The user should ensure that the Internet connection is stable for completing payment transaction in one go.
- The user should ensure that the payment transaction is completed within the shortest possible time limit. Otherwise, the session might expire and the money may remain unpaid to the e-commerce site, but debited from the bank account of the user.
- The user should ensure that the payment URL begins with https, which is a secure communication protocol over hyper text transmission protocol.
Payment gateway

▶ A third party online payment system which receives money from the buyer and transfers the same to the seller is called a payment gateway.

▶ CCAvenue, PaySeal, PayPal, etc. are some of the common payment gateways.

▶ For each successful transaction, the payment gateway takes a commission and transfers the remaining amount to the account of the beneficiary.
Cyber security

Cyber security

1. Protecting computer users from online attacks aimed at damaging the data, software, monetary loss or any other damages is called cyber security.

2. The term cyber means interaction of computers through Network.

3. Cyber crime is a malafide act happening inside the networking domain.

4. Victims of cyber crimes might lose data or money.
Information security

- Information security is the practice of preventing unauthorized use of information.
- Information Security is abbreviated to InfoSec.
- Under information security, any device having processor and memory is categorised under computing device.
- Keeping the data of a device safe from unauthorized users is called information security.
- CIA: CIA is the abbreviation for Confidentiality, Integrity & Availability under information security.
Threats to information security

- **Worms**: Worms are little scripts sent through email attachments. These enter the computer system when user tries to open the attachment. Worms might result in scare, loss of data from the system.

- **Phishing attacks**: Creating a cheat URL resembling an authentic website and making the victim to reveal login credentials is called phishing attack. In most of the cases, phishing victims innocently reveal bank login details, which are misused by the attacker.

- **Trojans**: A normal looking program may have an embedded malware. The malware might collect sensitive data and transmit the same to the attacker.

- **Computer virus**: A malicious which becomes hooked to the operating system and keeps continuously working to damage data stored in the device is called virus. Virus spreads through Internet, storage media like pen drive, etc.

- Breach of information security might result in theft of proprietary manufacturing processes or formulae from large firms.
Vulnerability

- Sources of weakness in the IT infrastructure for potential theft of information is called vulnerability.
- Vulnerabilities may be found in the operating system. These vulnerabilities are targeted by hackers and exploited.
- The network connection might be insecure due to lack of firewall, anti-virus and other online protection software packages.
- An application software might include a vulnerability. When this software runs, hackers may intrude the system and cause damages.
- Failure to conduct periodic security audit of computer systems might result in data vulnerability.
- Identifying sources of data disruption and ensuring continuous safety of business data is called risk management.
Risk management

- The likelihood of loss of data due to security vulnerability is called risk.
- Risk management is the process of ensuring smooth functioning of business without permitting any attack on the information held by an organization.
- A team of people calculate the risk faced by an organization based on existing vulnerabilities and sources of threat. The team decides upon the safe practices, software and hardware implementations to be adopted for complete information safety.
- To ensure information safety, an organization should have (i) security policy, (ii) organization of information security, (iii) human resource security, (iv) access control, (v) physical security, (vi) security incident management, (vii) safe disposal of damaged hard copies, (viii) safe disposal of damaged storage media.
Introduction to Directory Services

- Directory service is a single access point which could take a user to several other services located at different URLs.
- The directory service simplifies the job of security administrator to the safe maintenance of directory service alone.
- Directory service may store data in several computers. But, the user should find that the directory is a single point.
- The directory service should be able to include new objects based on changes to network.
- Searching for information from various locations on the network should be easy.
- The directory system should be independent of the physical location of the user or the administrator.
- The information stored in directory service should be accessible from any operating system.
- LDAP (Light Weight Directory Access Protocol) provides an easy mechanism for querying directory service.
Access control

- Access control is the selective permission to particular individuals to access a computer and perform their works.
- Access control may be physical security to the computer. Users may be permitted to use the computer only if they are authenticated through smart card, finger print, iris scan, etc.
- Access control may be implemented through software. Users may be required to provide a valid pair of identifier (ID) and password.
- Access control may be implemented through a combination of physical and software controls.
- The layout of the access control components is called access control topology.
Software development security

- Security is implemented at the development stage of a software.
- Security of software is designed to cover the entire life cycle of software.
- Software Development Life Cycle (SDLC) is the planning of the security measures to be incorporated in a software throughout the life span of the software.
- **Protection from disclosure**: is the prevention of disclosure of source code by any individual involved in the development of a software product.
- **Protection from alteration**: The source code should not be altered without appropriate authorization.
- **Protection from destruction**: Any individual should not have the right to delete or destroy the entire software.
- **Whois making the request**: Any request for changing or deleting software components should be recorded.
- **Verify access rights**: Any request from a user should be verified against the rights granted to the user. The user should be prevented from carrying out actions which are not permitted.
- **Maintaining historical records**: A chronological record of all accesses made by individuals and changes brought about to source code should be maintained.
- **Error handling**: The software framework should be able to handle configuration errors, session errors and exception in a consistent manner.
Privacy protection

- Data relating to several types of individuals may be held in the IT infrastructure of an organization.
- Protecting the data from unauthorized access is called privacy protection.
- Breach of privacy protection might result in huge damage to individuals and organizations.
- Medical records of patients, record of criminals, financial data of companies, genetic material, address and contact details of individuals, record of web surfing behaviour, record of credit card, debit card or Internet banking credentials are some of the privacy data which should not be compromised.
Information security audit

- Information security is audited by a team of experts with reference to vulnerabilities, threats and preventive measures for data protection.
- The information security audit thoroughly analyses all threat events of the past and formulates security measures for the future.
- Information security audit should be performed in a periodic manner, i.e., once every month, every quarter, etc.
- Segregation of duties should be adopted to ensure that no single employee has all the data needed to make a fraudulent transaction.
- All segments of the organization should be protected through physical and password security measures.
- The audit team should ensure that customers of the organization can feel safe when their data is divulged for business purpose.
- IT audit should probe abnormal behaviour of any individual, which might be an indication of potential fraud/theft of data.
Introduction to I.T. Act

- Information Technology Act, 2000 was introduced to prevent and punish cyber crimes.
- I.T. Act provides support for ecommerce.
- I.T. Act protects individuals from being victimized by cyber frauds.
- A major amendment to the IT act was made in 2008.
# Penalties for cybercrimes

## Table: Cyber crimes and punishments

<table>
<thead>
<tr>
<th>No.</th>
<th>Section</th>
<th>Cyber crime</th>
<th>Max. Imprisonment</th>
<th>Max. Fine</th>
<th>Both imp' and fine?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>65</td>
<td>Tampering with computer source documents</td>
<td>2 years</td>
<td>2Lakh</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>66</td>
<td>Hacking</td>
<td>3 years</td>
<td>5Lakh</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>66B</td>
<td>Receiving stolen computer/ mobile</td>
<td>3 years</td>
<td>1Lakh</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>66C</td>
<td>Misuse of password</td>
<td>3 years</td>
<td>1Lakh</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>66D</td>
<td>Cheating with computer</td>
<td>3 years</td>
<td>1Lakh</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>66E</td>
<td>Publishing private images of others</td>
<td>3 years</td>
<td>2Lakh</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>66F</td>
<td>Cyber terrorism</td>
<td>Life</td>
<td>—</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>67</td>
<td>Publishing obscene content</td>
<td>5 years</td>
<td>1Lakh</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>67A</td>
<td>Publishing sexual content</td>
<td>7 years</td>
<td>1Lakh</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>67B</td>
<td>Publishing child porn</td>
<td>5 years (first); 7 years (subsequent)</td>
<td>1Lakh (first); 1Lakh (subsequent)</td>
<td>Yes</td>
</tr>
</tbody>
</table>
https://www.w3schools.com/js/js_htmlldom.asp


https://en.wikipedia.org/wiki/Visual_Basic_for_Applications