Q.2 a. What do you mean by command line arguments? Write a program that displays the command line arguments entered by the user. (3+5)

Answer

Command line arguments are parameters that are supplied to the application program at the time of invoking it for execution. We can write Java programs that can receive and use the arguments provided in the command line. The signature of the **main()** method used in the program is as follows:

```
public static void main(String args[])
```

Here args is declared as an array of strings (known as string objects). Any arguments provided in the command line (at the time of execution) are passed to the array **args** as its elements. We can simply access the array elements and use them in the program as we wish. For example, consider the command line

```
java Test BASIC FORTRAN C C++
```

This command line contains four arguments, which are assigned the array **args** as follows:

```
BASIC      args[0]
FORTRAN    args[1]
C          args[2]
C++        args[3]
```

The individual elements of an array are accessed by using an index or subscript like **args[i]**. The value of i denotes the position of the elements inside the array. For example **args[2]** denotes the third element and represents C.

The following program uses command line arguments as input.

```
public class CommandLine {

    public static void main (String args[]) {
      int count, i=0;
      String str;
      count = args.length;
      System.out.println("Number of arguments = " + count);

      while (i < count){
            str = args[i];
            System.out.println("args[“+i+”] = " + str);
            i++;
      }
    }
}
```

When the program is compiled and run with the command as:

java CommandLine BASIC FORTRAN C C++

the output of the program would be as follows:

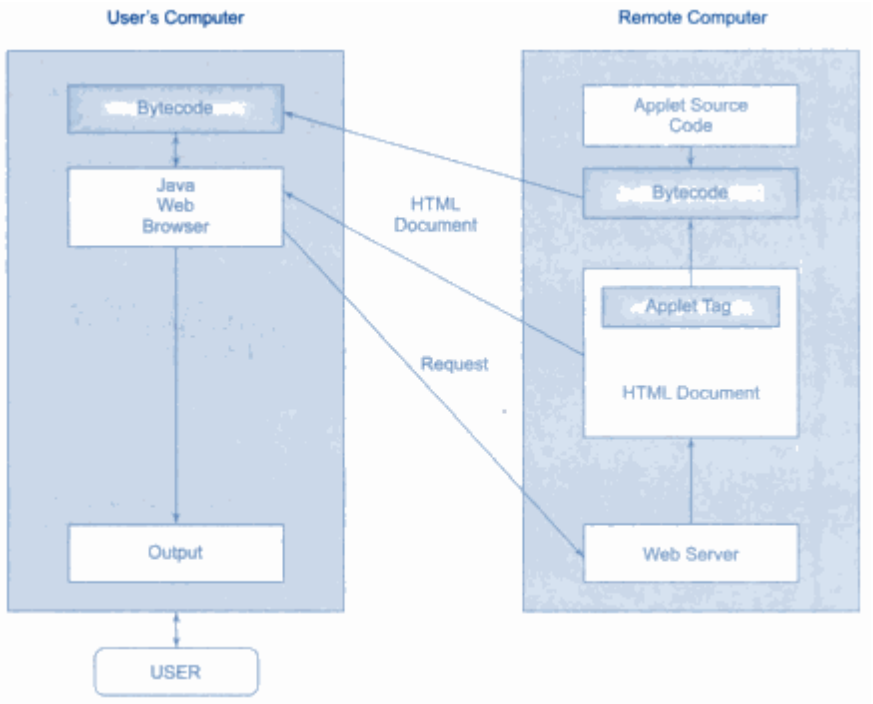
```
Number of arguments = 4
args[0] = BASIC
args[1] = FORTRAN
args[2] = C
args[3] = C++
```

b. With the help of a suitable figure, describe the different steps using which Java communicates with a web page. (8)

Answer

Java communicates with a web page through a special tag called <APPLET>. The following figure illustrates the process. The figure shows the following communication steps:

- The user sends a request for an HTML document to the remote computer's Web server. The Web server is a program that accepts a request, processes the request, and sends the required document.
- The HTML document is returned to the user's browser. The document contains the APPLET tag, which identifies the applet.
- The corresponding applet bytecode is transferred to the user's computer. This bytecode had been previously created by the Java compiler using the Java source code file for that applet.
- The Java-enabled browser on the user's computer interprets the bytecode and provides output.
- The user may have further interaction with the applet but with no further downloading from the provider's Web server. This is because the bytecode contains all the information necessary to interpret the applet.



Java's interaction with the web

Q.3 a. Explain the syntax of *for* loop. Also, describe with the help of example, the *enhanced for* loop. (4+4)

Answer

The ***for*** loop is entry controlled loop that provides a more concise loop control structure. The syntax of the *for* loop is as follows:

```
for (initialization; test_condition; increment) {
      // body of loop
}
```

The execution of the ***for*** statement is as follows:

1. *Initialization* of control variables is done first, using assignment statements such as i=1. The variable **i** is known as loop-control variable.
2. The value of the control variable is tested using the *test_condition*. The test condition is a relational expression, such as i < 10 that determines when the loop will exit. If the condition is true, the body of the loop will executed; otherwise the loop is terminated and the execution continues with the statement that immediately follows the loop.
3. When the body of the loop is executed, the control is transferred back to the **for** statement after evaluating the last statement in the loop. Now, the control variable is *incremented* using

assignment statement such as *i = i + 1* (or *i++*) and the new value of the control variable is again tested to test whether it satisfies the loop condition. It the condition satisfied, the body of the loop is again executed. This process continues till the value of the control variable fails to satisfy the test condition.

Consider the following program segment:

```
for (i = 0; i < 10; i++) {
      System.out.println(i);
}
```

This **for** loop is executed 10 times and prints the digit 0 to 9 in one line. The three sections enclosed within parenthesis must be separated by semicolons. But there is no semicolon at the end of the increment section.

The **for** statement allows for *negative increments*. For example, the loop can be written as:

```
for (i = 9; i >= 0; i=i-1)
      System.out.println(i);
```

This loop is also executed for 10 times, but the output would be from 9 to 0. The braces are optional when the body of the loop contains only one statement.

Since the conditional test is always performed at the beginning of the lop, the body of the loop may not be executed at all, if the condition fails at the start. For example,

```
for (i = 10; i < 10; i++) {
      System.out.println(i);
}
```

will never be executed because the test condition fails at the very beginning itself.

Enhanced for loop

The enhanced **for** loop, also called *for each* loop, is an extended language feature introduced with the J2SE 5.0 release. This feature helps us to retrieve the array elements efficiently rather than using array indexes. We can also use this feature to eliminate the iterators in a for loop and to retrieve the elements from a collection. The enhanced for loop takes the following form:

```
for (Type Identifier : Expression) {
      // statements;
}
```

where, *Type* represents the data type or object used; *Identifier* refers to the name of a variable; and *Expression* is an instance of the java.lang.Iterable interface or an array.

For example, consider the following statements:

```
int numArr[3] = {56, 48, 79};
for (int i=0; i<3; i++) {
      if (numArr[i]>50 && numArr[i]<100) {
            System.out.println("Value is " + numArr[i]);
      }
}
```

which is equivalent to the following code:

```
int numArr[3] = {56, 48, 79};
for (int i : numArr) {
      if (i>50 && i<100) {
            System.out.println("Value is " + i);
      }
}
```

Thus, we can use the enhanced for loop to track the elements of an array efficiently. In the same manner, we can track the collection elements using the enhanced for loop as follows:

```
Stack ss = new Stack();
ss.push(new Integer(56));
ss.push(new Integer(48));
```

```
ss.push(new Integer(79));
for(Object ob : ss) {
      System.out.println(ob);
}
```

b. Write a program to find the number of and sum of all integers greater than 100 and less than 200 that are divisible by 7. (8)

Answer

```
public class SumOfNumber {

    public static void main(String args[]) {
      int i;
      int sum = 0;
      int count = 0;
      System.out.println("Program to find the sum of all numbers
      between 100 and 200 which are divisible by 7");

      for (i=100; i<200; i++){
            if ( i % 7 == 0){

                  sum = sum + i;
                  count++;
            }
      }
      System.out.println("Number of INTEGERS divisible by 7 are :
            " + count);
      System.out.println("Required Sum is : " + sum);
    }
}
```

Q.4 a. What do you mean by *'final method'* and *'final class'*? Can a class be declared *final* and *abstract* at the same time? Support your answer.(3+3+2)

Answer

Final method is used to Prevent Overriding

While method overriding is one of Java's most powerful features, there will be times when you will want to prevent it from occurring. To disallow a method from being overridden, specify **final** as a modifier at the start of its declaration. Methods declared as **final** cannot be overridden. The following fragment illustrates **final**:

```
class A {
      final void meth() {
            System.out.println("This is a final method.");
      }
}

class B extends A {
      void meth() { // ERROR! Can't override.
            System.out.println("Illegal!");
      }
}
```

Because **meth( )** is declared as **final**, it cannot be overridden in **B**. If you attempt to do so, a compile-time error will result.

Methods declared as **final** can sometimes provide a performance enhancement: The compiler is free to *inline* calls to them because it "knows" they will not be overridden by a subclass. When a small **final** method is called, often the Java compiler can copy the bytecode for the subroutine directly inline with the compiled code of the calling method, thus eliminating the costly overhead associated with a method call. Inlining is only an option with **final** methods. Normally, Java resolves calls to methods dynamically, at runtime. This is called *late binding.* However, since **final** methods cannot be overridden, a call to one can be resolved at compile time. This is called *early binding.*

Final class is used to Prevent Inheritance

Sometimes you will want to prevent a class from being inherited. To do this, precede the class declaration with **final**. Declaring a class as **final** implicitly declares all of its methods as **final**, too. As you might expect, it is illegal to declare a class as both **abstract** and **final** since an abstract class is incomplete by itself and relies upon its subclasses to provide complete implementations.

Here is an example of a **final** class:

```
final class A {
      // ...
}

// The following class is illegal.
class B extends A { // ERROR! Can't subclass A
      // ...
}
```

As the comments imply, it is illegal for **B** to inherit **A** since **A** is declared as **final**.

NO, a class can't be declared *final* and *abstract* at the same time.

Reason

Final class means that class can't be further subclassed. But when we declared a class as **abstract** it only contains the method declaration and their definition are to be implemented in the subclass (or subclasses). Hence the class can't be declared as **final** and **abstract** at the same time.

b. Given are two one-dimensional integer arrays A and B in which numbers are stored in ascending order. Write a program to merge them into a single sorted array C that contains every item from arrays A and B, in ascending order. (8)

Answer

```
import java.util.*;

public class MergingOfTwoArray {

    public static void main(String args[]) {
      int arr1[] = new int[10];
      int arr2[] = new int[10];
      int arr3[] = new int[20];
      int k;
            int l;
            int m;

      System.out.println("Program to MERGE two SORTED array into
            third array in ASCENDING order");

            Scanner s = new Scanner(System.in);
            System.out.println("Enter FIRST Array");
```

```
            for (int i=0; i<arr1.length; i++)
                  arr1[i] = (s.nextInt());

            System.out.println("Enter Second Array");
            for (int i=0; i<arr2.length; i++)
                  arr2[i] = (s.nextInt());

            k=0;
            l=0;
            m=0;
            while(l<10 && m<10){
                  if (arr1[l] < arr2[m]){
                        arr3[k] = arr1[l];
                        l++;
                  }
                  else {

                        arr3[k] = arr2[m];
                        m++;
                  }
                  k++;
            }

            while(l < 10){
                  arr3[k] = arr1[l];
                  k++;
                  l++;
            }

            while(m < 10){
                  arr3[k] = arr2[m];
                  k++;
                  m++;
            }

            System.out.println("Merged Array is :");
            for(int i=0; i<20; i++){
                  System.out.print(arr3[i] + " ");
            }

    }
}
```

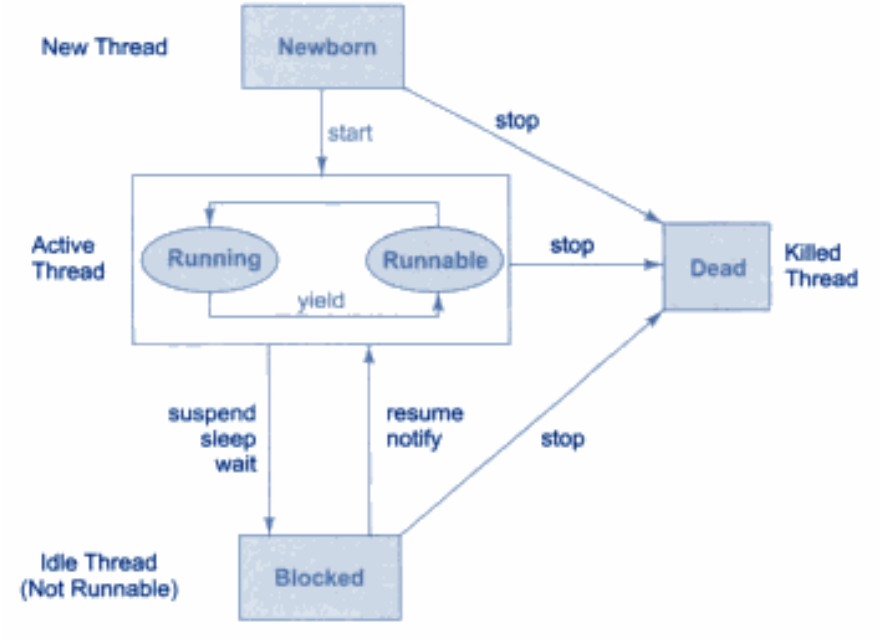
Q.5 a. Describe the life cycle of a thread with the help of a figure. (5+3)

Answer

During the life time of a thread, there are many states it can enter. They include:

- Newborn state
- Runnable state
- Running state
- Blocked state
- Dead state

A thread is always in one of these five states. It can move from one state to another via a variety of ways as shown in following figure:



State transition diagram of a thread

Newborn state

When we create a thread object, the thread is born and is said to be in newborn state. The thread is not yet scheduled for running. At this state, we can do only one of the following things with it:

- Schedule it for running using **start()** method.
- Kill it using **stop()** method.

If scheduled, it moves to the runnable state. If we attempt to use any other method at this stage, an exception will be thrown.

Runnable state

The runnable state means that the thread is ready for execution and in waiting for the availability of the processor. That is, the thread has joined the queue o0f threads that are waiting for execution. If all threads have equal priority, then they are given equal time slots for execution in round robin fashion, i.e., first-come, first-serve manner. The thread that relinquishes control joins the queue at the end and again waits for its turn. This process of assigning time to threads is known as time-slicing.

But, if we want a thread to relinquish control to another thread to equal priority before its turn comes, we can do so by using the **yield()** method.

Running state

Running means that the processor has given its time to the thread for its execution. The thread runs until it relinquishes control on its own or it is pre-empted by a higher priority thread. A running thread may relinquish its control in one of the following situations:

1. It has been suspended using **suspend()** method. A suspended thread can be revived by using the **resume()** metho0d. This approach is useful when we want to suspend a thread for sometime due to certain reasons, but do not want to kill it.
2. It has been made to sleep. We can put a thread to sleep for a specified time period using the method **sleep(time)** where time is in milliseconds. This means that the thread is out of the queue during this time period. The thread re-enters the runnable state as soon as this time period is elapsed.
3. It has been told to wait until some event occurs. This is done using the **wait()** method. The thread can be scheduled to run again using the **notify()** method.

Blocked state

A thread is said to be blocked when it is prevented from entering into the runnable state and subsequently the running state. This happens when the thread is suspended, sleeping, or waiting in order to satisfy certain conditions. A blocked thread is considered "not runnable" but not dead and therefore fully qualified to run again.

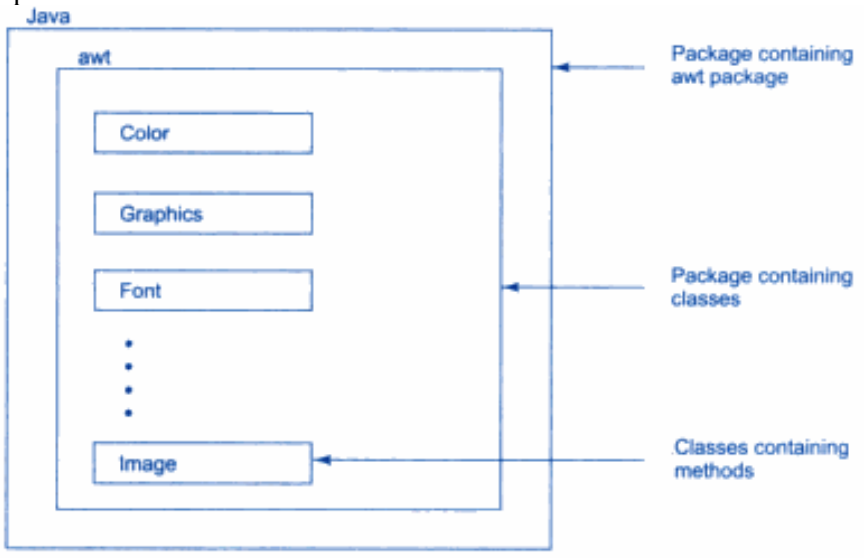
Dead state

Every thread has a life cycle. A running thread ends its life when it has completed executing its **run()** method. It is a natural death. However, we can kill it by sending the stop message to it at any state thus causing a premature death to it. A thread can be killed as soon as it is born, or while it is running, or even when it is in "not runnable" condition.

b. Discuss, with the help of examples, ways of accessing the classes stored in a package using import statement and without import statement. (8)

Answer

The packages are organised in a hierarchical structure as illustrated in following figure. This shows that the package named **java** contains the package **awt,** which in turn contains various classes required for implementing graphical user interface.



Hierarchical representation of java.awt package

There are two ways of accessing the classes stored in a package. The forst approach is to use the *fully qualified class name* of the class that we want to use. This is done by using the package name containing the class and then appending the class name to it using dot operator. For example, if we want to refer to the class **Color** in the **awt** package, then we may do so as follows:

```
java.awt.Color
```

where **awt** is a package within the package **java** and the hierarchy is represented by separating the levels with dots. This approach is perhaps the best and easiest one if we need to access the class only once or when we need not have to access any other classes of the package.

But in many situations, we might want to use a class in a nuber of places in the program or we may like to use many of the classes contained in a package. We may achieve this easily as follows:

```
import packagename.ClassName;
      OR
import packagename.*;
```

These are known as *import statements* and must appear at the top of the file, before any class declarations, **import** is a keyword.

The first statement allows the specified class in the specified package to be imported. For example, the following statement imports the class **Color** and therefore the class name can now be directly used in the program. There is no need to use the package name to qualify the class.

```
import java.awt.Color;
```

The second statement imports every class contained in the specified package. For example the following statement imports all classes of **java.awt** package.

```
import java.awt.*;
```

Q.6 a. What is an exception? Discuss the basic concepts of exception handling. List three common types of exceptions that might occur in Java. (2+3+3)

Answer

An exception is a condition that is caused by a run-time error in the program. When the Java interpreter encounters an error such as dividing an integer by zero, it creates an exception object and throws it (i.e. informs us that an error has occurred).

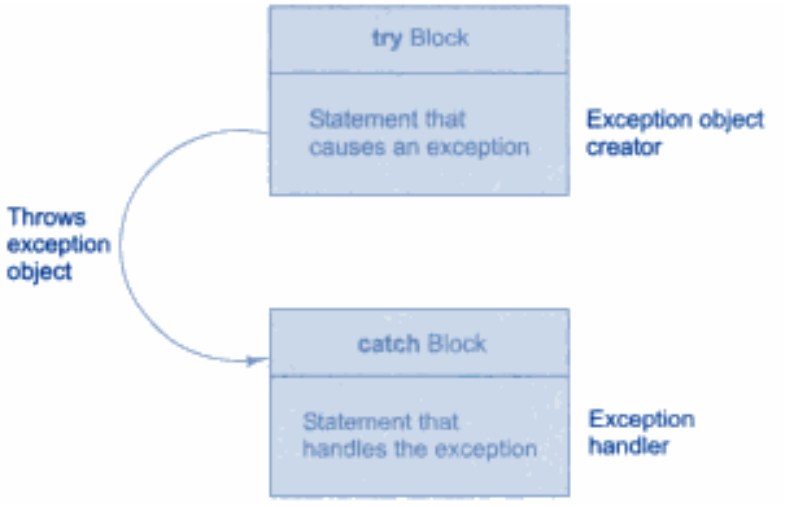
If the exception object is not caught and handled properly, the interpreter will display as error message and will terminate the program. If we want the program to continue with the execution of the remaining code, then we should try to catch the exception object thrown by the error condition and then display the appropriate message for taking corrective actions. This task is known as ***exception handling***.

The purpose of exception handling mechanism is to provide a means to detect and report an "exceptional circumstance" so that appropriate action can be taken. The mechanism suggests incorporation of a separate error handling code that performs the following tasks:

- Find the problem (***Hit*** the exception).
- Inform that an error has occurred (***Throw*** the exception).
- Receive the error information (***Catch*** the exception).
- Take corrective actions (***Handle*** the exception).

The error handling code basically consists of two segments, one to detect errors and to throw exceptions and the other to catch exceptions and to take appropriate actions.

The basic concepts of exception handling are throwing an exception and catching it as illustrated in the following figure:



Exception handling mechanism

Java uses a keyword **try** to preface a block of code that is likely to cause an error condition and "throw" an exception. A catch block defined by the keyword **catch** "catches" the exception "thrown" by the try block and handles it appropriately. The catch block is added immediately after the try block. The following syntax illustrates the use of simple **try** and **catch** statements:

```
.................
.................
try {
      ..........
      statement        // generates an exception
      ....
} catch(exception.type e) {
      Statement;       // processes the exception
}
.................
.................
```

The try block can have one or more statements that could generate an exception. If anyone statement generates an exception, the remaining statements in the block are skipped and execution jumps to the catch block that is placed next to the try block.
The catch block too can have one or more statements that are necessary to process the exception. Every **try** block should be followed by *at least* one catch block; otherwise compilation error will occur.
The **catch** block works like a method definition. The **catch** block is passed a single parameter, which is reference to the exception object thrown; by the try block. If the catch parameter matches with the type of the exception object, then the exception is caught and statements in the catch block will be executed. Otherwise, the exception is not caught and the default exception handler will cause the execution to terminate.

Some common Java exceptions are as follows:

1	ArithmeticException	Caused by math errors such as division by zero
2	ArrayIndexOutOfBoundException	Caused by bad array indexes
3	FileNotFoundException	Caused by an attempt to access a nonexistent file
4	IOException	Caused by general I/O failures, such as inability to read from a file
5	NumberFormatException	Caused when a conversion between strings and number fails

b. Write a program that generates random integers and stores them in a file named "rand.dat". The program then reads the integers from file and display on the screen. (8)

Answer

```
import java.io.*;
public class TestReadWriteIntegers {

    public static void main (String args[]) {
      DataInputStream dis = null;
      DataOutputStream dos = null;

      File intFile = new File("rand.dat");

      try {
            dos = new DataOutputStream(new
                        FileOutputStream(intFile));
            for (int i=0; i<20; i++)
                  dos.writeInt((int)(Math.random()*100));
      }
```

```
        catch (IOException ioe){
              System.out.println(ioe.getMessage());
        }
        finally{
              try {
                    dos.close();
              } catch (IOException ioe){
              }

        }

        try {
              dis = new DataInputStream(new FileInputStream
                    intFile));
              for (int i=0; i<20; i++){
                    int n = dis.readInt();
                    System.out.print(n + " ");
              }
        } catch (IOException ioe) {
                    System.out.println(ioe.getMessage());
              } finally {
                    try {
                          dis.close();
                    } catch (IOException ioe) {
              }
              }

     }


}
```

PART B
Attempt any TWO questions. Each question carries 16 marks

Q.7 a. Discuss the general syntax rules of XHTML. List and explain the core attributes for XHTML. (4+4)

Answer

The syntax or grammar of HTML (XHTML) is actually defined using SGML (Standard Generalized Markup Language). SGML is an ISO (International Standards Organization) standard formal language for defining markup languages. HTML is defined with a SGML DTD that specifies available elements such as head, body, h1, p, and so on. By revising and extending the DTD, HTML can be evolved with relative ease. The three flavors of XHTML 1.0 are supported by different DTDs.

The following general syntax rules will help you use XHTML:

- All tags begin with < and ends with >. The tag name is given immediately following the leading <. Make sure the tag is spelled correctly. Unrecognized tags are ignored by browsers. Any attributes are given following the tag name in the form:
  <tag attribute1 ="value" attribute2 ="value" ... >
- Tag and attribute names are given in lower case. Attributes are always given in the form
  attributeName="value"
  where the value is case sensitive.

- Unrecognized tags and attributes are ignored by browsers.
- Most elements involve open and close tags. Other elements, such as
, and (inline image), do not have closing tags and are known as empty elements.
  Note the use of the extra space in front of `/' for empty elements.
- Elements must be well-formed. It means no missing begin or end tags and no improper element nesting. For example,
  Learning XHTML
  overlaps the tags and is not properly nested. Existing browsers may tolerate such ill-formed code. The correct nesting is
  Learning XHTML
- Attributes can be required or optional and can be given in any order. If an attribute is not given its default value, if any, is used.
- Extra white space and line breaks are allowed between the tag name and attributes, and around the = sign inside an attribute. Line breaks and whitespace within attribute values are also allowed but should be avoided because they may be treated inconsistently by browsers.
- The body element may contain only block-level HTML elements. Freestanding texts (not enclosed in block elements) or inline elements are not allowed directly in the body element.

Certain tags are only allowed within their permitted context. For example, a <tr> (table row) element can only be given inside a <table> element. Learning XHTML involves knowing the elements, their attributes, where they can be placed, and the elements they can contain.

All XHTML elements admit the following core attributes:

- id: Uniquely identifies the element in a page. All ids in a document must be distinct. Among other uses, a URL ending in #some id can lead directly to an element inside a document.
- style: Gives presentation styles for the individual element. For example, the code
  <body style="background-color: cyan">
  gives the color value cyan to the style property background-color for this element. Several style properties separated by semicolons (;) can be given. The style attribute is a direct but inflexible way to specify presentation style. While this attribute is sometimes necessary, better and much more versatile methods for assigning styles.
- class: Specifies the style class for the element. Thus, you may place HTML elements in different classes and associate presentation styles to all elements belonging to the same class.
- title: Provides a title for the element. This may be used for tool-tip display by browsers.

b. What are name servers? What are the three elements of DNS? (2+6)

Answer

Name servers are the actual programs that provide the domain-to-IP mapping information on the Internet. DNS provides a distributed database service that supports dynamic retrieval of information contained in the name space. Web browsers, and other Internet client applications, will normally use the DNS to obtain the IP a target host before making contact with a server.

There are three elements to the DNS: the name space, the name servers, and the resolvers.

- **Name servers**: Information in the distributed DNS are divided into zones and each zone is supported by one or more name servers running on different hosts. A zone is associated with a node on the domain tree and covers all or part of the subtree at that node. A name server that has complete information for a particular zone is said to be an authority for that zone. Authoritative information is automatically distributed to other name servers which provide redundant service for the same zone. A server relies on lower-level servers for other information within its sub-domain and on external servers for other zones in the domain tree. A server associated with the root node of the domain tree is a root server and can lead to information anywhere in the DNS. An authoritative server uses local files to store information, to locate key servers within and

without its domain, and to cache query results from other servers. A boot file, usually /etc/named.boot, configures a name server and its data files.

The management of each zone is also free to designate the hosts that run the name servers and to make changes in its authoritative database. For example, the host ns.nic.ddn.mil may run a root name server. The host condor.mcs.kent.edu may run a name server for the domain mcs.kent.edu.

A server answers queries from resolvers and provides either definitive answers or referrals to other name servers. The DNS database is set up to handle network address, mail exchange, host configuration, and other types of queries, some yet to be implemented.

- Resolvers: A DNS resolver is a program that sends queries to name servers and obtains replies from them. On UNIX systems, a resolver usually takes the form of a C library function. A resolver can access at least one name server and use that name server's information to answer a query directly, or pursue the query using referrals to other name servers.

  Resolvers, in the form of networking library routines, are used to translate domain names into actual IP addresses. These library routines, in turn, ask prescribed name servers to resolve the domain names. The name servers to use for any particular host are normally specified in the file /etc/resolv.conf or /usr/etc/resolv.conf.

The ICANN maintains root name servers associated with the root node of the DNS tree. Domain name registrars, corporations, organizations, Web hosting companies, and other Internet Service Providers (ISP) run name servers for their zones to associate IPs to domain names in their particular zones. All name servers on the Internet cooperate to perform domain-to-IP mappings on-the-fly.

Q.8 a. With the help of HTML code, discuss Page Forwarding. (2+2)

Answer

Web pages sometimes must move to different locations. But visitors may have bookmarked the old location or search engines may still have the old location in their search database. When moving Web pages, it is prudent to leave a forwarding page at the original URL, at least temporarily.

A forwarding page may display information and redirect the visitor to the new location automatically. Use the meta tag

```
<meta http-equiv="Refresh" content="8; url=newUrl " />
```

The http-equiv meta tag actually provides an HTTP response header for the page. The above meta element actually give the response header

```
Refresh 8; url=newUrl
```

The effect is to display the page and load the newUrl after 8 seconds. Here is a sample forwarding page

```
<head><title>Page Moved</title>
     <meta http-equiv="Refresh" content="8; url=target_url" />
</head><body>
<h3>This Page Has Moved</h3>
     <p>New Web location is: ... </p>
     <p>You will be forwarded to the new location automatically.</p>
</body></html>
```

The Refresh response header (meta tag) can be used to refresh a page periodically to send updated information to the end-user. This can be useful for displaying changing information such as sports scores and stock quotes. Simply set the refresh target url to the page itself so the browser will automatically retrieve the page again after the preset time period. This way, the updated page, containing the same meta refresh tag, will be shown to the user.

b. What is table in XHTML? Explain any three different attributes for the table element that controls the various formatting options for the table. (3+3)

Answer

Tables let us display information in a clear and concise fashion. The block element table organizes and presents information in neatly aligned rows and columns. It is often used to

- present tabular data
- layout Web pages by aligning content and graphics in horizontal and vertical grids to achieve visual communication designs
- organize entries in fill-out forms for user input

In general, a table involves many aspects: rows, columns, data cells, headings, lines separating cells, a caption, spacing within and among cells, and vertical and horizontal alignments of cell contents. The rows and columns can be separated into groupings. A cell can also span several columns or rows. Thus, table is a complicated HTML construct and it takes some doing to understand and master. But, the ability to use table effectively is critical for HTML programming.

Attributes for the table element control various formatting options for the table.

- Border: specifies the width in pixels of the border around a table . A non-zero border setting also implies horizontal and vertical rules, and a zero value also implies no rules.
- Rules: specifies the thin 1-pixel rules separating table cells: none (no rules), groups (rules between row groups and column groups only), rows (rules between rows only), cols (rules between columns only), and all (rules between all cells). The default is none if border is not set (set to 0) and the default is all if border is set. This attribute is poorly supported by browsers.
- Frame: specifies the visible sides of the table's outer border: void (no side), above/below (top/bottom), vsides/hsides (left+right or top+bottom), lhs/rhs (left or right), border (all sides).
- Cellspacing: specifies the amount of space between table cells . The default cellpacing is usually 1 pixel.
- Cellpadding: defines the amount of space between the cell border and cell contents . The default cellpadding is usually 1 pixel.
- Summary: text describing the purpose and content of the table for non-visual browsers.

c. What is layout grid? What are the basic layout strategies generally designers use in designing a web page (any three)? (3+3)

Answer

A grid is a set of invisible vertical and horizontal lines to guide page layout. It is the primary way designers organize elements in a two-dimensional space. A grid aligns page elements vertically and horizontally, marks margins, and sets start and end points for element placement. A well-designed grid makes a page visually clear and pleasing, resulting in increased usability and effective content delivery. A consistent page layout also helps to create unity throughout the site. Grids are certainly not a new invention. They have been used for centuries as the basis for ornamental design in screens, textiles, quilt design architecture.

Designers generally have used these basic layout strategies:

- Left-justified layouts: a fixed-width grid is used and the page starts at the left margin. This design can leave an annoying white space on the right side for larger resolution screens.
- Centered layout: a fixed-width grid is centered horizontally in the browser window.
- Full-width fluid layout: the grid is scalable and fills the width of the browser window and responds to browser resizing dynamically. For larger screens, text lines can become too long for easy reading.
- Centered fluid layout: the layout uses fluid but equal left and right margins and a fluid centered grid.

Q.9 a. Explain the process of deploying CGI programs. (5)

Answer

A CGI program is deployed by placing it in the cgi-bin directory designated by the Web server. The cgi-bin directory should allow Web access:

```
chmod o+rx cgi-bin
```

And any CGI program placed in this directory must also allow Web access:

```
chmod o+rx program.cgi
```

When deploying a Perl program, make sure the first line of the program

```
#!/usr/bin/perl
```

indicates the location of the perl command on that particular host. If you copy-and-paste a Perl script, make sure you delete and retype the first line using a text editor on the host computer. Hidden characters on the first line have caused many a headache for the unwary.

More likely than not, a Perl CGI program will use the CGI.pm perl module with the line use CGI qw(:standard);

For this to work you need to be sure that Perl on your computer has that module installed in a location included on the Perl variable @INC which is a list of directories where Perl headers and modules are found. If you are installing a Perl script on a host meant to serve the Web, all this are most likely already set.

Once deployed, the URL to reach the CGI program is in the general form

http://host/cgi-bin/program.cgi

b. Write the advantages of using cascading style sheet in web page design. (5)

Answer

Following are the advantages of using Cascading style sheet in web page design:

- **Separation of content from presentation**
  CSS facilitates publication of content in multiple presentation formats based on nominal parameters. Nominal parameters include explicit user preferences, different web browsers, the type of device being used to view the content (a desktop computer or mobile Internet device), the geographic location of the user and many other variables.
- **Site-wide consistency**
  When CSS is used effectively, in terms of inheritance and "cascading," a global style sheet can be used to affect and style elements site-wide. If the situation arises that the styling of the elements should need to be changed or adjusted, these changes can be made by editing rules in the global style sheet. Before CSS, this sort of maintenance was more difficult, expensive and time-consuming.
- **Bandwidth**
  A stylesheet, internal or external, will specify the style once for a range of HTML elements selected by class, type or relationship to others. This is much more efficient than repeating style information inline for each occurrence of the element. An external stylesheet is usually stored in the browser cache, and can therefore be used on multiple pages without being reloaded, further reducing data transfer over a network.
- **Page reformatting**
  With a simple change of one line, a different style sheet can be used for the same page. This has advantages for accessibility, as well as providing the ability to tailor a page or site to different target devices. Furthermore, devices not able to understand the styling still display the content.
- **Accessibility**
  Without CSS, web designers must typically lay out their pages with techniques that hinder accessibility for vision-impaired users, like HTML tables

c. Explain how JavaScript are embedded in a web page. (6)

Answer

A Javascript program usually consists of various code parts that cooperate to do the job. Where and how to place code in a Web page depend on the purpose of the code. But generally, the code can appear either in <script> elements or as values of event-handling attributes of HTML tags. Any number of script elements can be placed in head, body, block and inline elements. Place a script in the head element unless it generates document content, as in Ex: Date.

- Code for defining functions or creating data structures are placed in <script> elements inside the page's <head> element.
- Code for generating HTML to be displayed as part of the page are placed in <script> elements inside the <body> where the generated code will be inserted.
- Code for actions triggered by events are given as values of event attributes of HTML tags. The code is usually a function call or a few short statements. The onfocus, onblur, onclick, onmouseover, and onmouseout are examples of event attributes.

A Javascript program often works for not one but a set of pages. In that case, it is better to place the code in a separate file. The file can then be attached to a Web page as follows:

```
<script type="text/javascript" src="file.js"></script>
```

The src attribute of <script> gives the URL of the program file. Here we used the conventional .js file name suffix.

With file inclusion, any corrections or improvements in the .js file will be reflected in multiple pages automatically. Furthermore, a browser will download the program only once, reducing download time for these pages. Older browsers that do not support client-side scripting will ignore the <script> tag and the program file will not be included. If the code is placed in the <script> tag, then the following construct is often used to avoid the code being mistaken as page content by older browsers:

```
<script type="text/javascript">
<!--
var d = new Date;
...
// -->
</script>
```

TEXT BOOK

I. Programming with Java – A Primer, E. Balagurusamy, Third Edition, TMH, 2007

II. An Introduction to Web Design + Programming, Paul S. Wang and Sanda S. Katila, Thomson Course Technology, India Edition, 2008