1. a) Name the header files needed to compile the C++ program given below: [2]

```
void main()
{
   char mm[12]="SepTEmBer"; int dd=1+random(30);
   printf("Month: %s , Days:%i\n", strupr(mm), dd);
   getch();
}
```

<stdlib.h> for random(), <stdio.h> for printf(), <string.h> for strupr(), <conio.h> for getch()

b) Rewrite the program after removing all the syntax errors. **Underline** the corrections. [3]

```
#include stdio.h
struct printer
{
   char model[30]; double price; int qty;
}
void main[]
{
   printer p;
   cout<<'Model? '; gets(p.model);
   cout<<"Price Quantity? "; cin>>price>>p.qty;
   print("%s %f %i\n", p.model, p.price, p.qty);
}
```

Corrected Program is given below:

```
#include<iostream.h>
#include<stdio.h>
struct printer
{
   char model[30]; double price; int qty;
};
void main()
{
   printer p;
   cout<<"Model? "; gets(p.model);
   cout<<"Price Quantity? "; cin>>p.price>>p.qty;
   printf("%s %f %i\n", p.model, p.price, p.qty);
}
```

c) Give the output of the programs given below: [4]

i)

```
#include<iostream.h>
struct metro { char city[20]; double cel; };
void mf(metro &x, char c[]) { strcpy(x.city, c); x.cel+=7.2; }
void mf(metro &y, double t) { y.cel=+t; }
void mf(metro &z) { z.cel=+3.5; }
void mshow(metro m) { cout<<m.city<<","<<m.cel<<endl; }
void main()
{
   metro a={"KOLKATA", 10.7};
   metro b={"NEW DELHI"};
   mf(b, "MUMBAI"); mf(a, 8.5);
   mshow(a);
   mshow(b);
   mf(b, 2.8); mf(a);
   mshow(b);
   mshow(a);
}
```

```
KOLKATA,8.5 / 19.2
MUMBAI,7.2 / 7.2
MUMBAI,2.8 / 10
KOLKATA,3.5 / 22.7
```

ii)
```
#include<iostream.h>
void main()
{
   int a[8]={27, 37, 43, 58, 74, 63, 94, 86};
   for (int x=0; x<8; x++)
      cout<<(a[x]/10+a[x]%10*10)<<'$';
}
```
```
72$73$34$85$47$36$49$68$
```

3. a) Define string. Differentiate between **strcpy**() and **strcat**() with suitable C++ code. [2]

A string is an array of characters terminated by a nul characters.
Function strcpy() duplicates (copies) a string and function strcat() joins two strings.

```
char a[20]="MANGAF", b[20];
strcpy(b, "KUWAIT");     //duplicates "KUWAIT" to b
strcat(a, b);            //joins b after a
cout<<a<<","<<b<<endl;   //displays MANGAF,MANGAFKUWAIT
```

b) What is the drawback of inputting a string using **cin**? How to overcome this drawback? [1]

Using cin, one can input string without any space or tab. To input a string that may contain either space or tab, we should use gets() from the header file <stdio.h>

c) What is an open array? Write C++ code showing the two ways of using an open array. [2]

An open array is array without any size. Generally an array cannot be created without any size but in a C++ program we can create open array in possible ways:

- An open array is created with an initializer
```
double a[]={23.5, 45.6, 71.8, 87.4, 68.3};
char b[]="AHMADI, KUWAIT";
```
- An open array as a formal parameter to a function
```
void display(int a[], int n)
{
   for (int k=0; k<n; k++)
      cout<<a[k]<<' ';
}
```

d) What is a subscript of an array? What is a subscripted variable? [1]

Subscript (index) is an integer value, starting from 0 assigned to every element of an array. An array element is called subscripted variable because an element of array contain array name and its subscript.

4. a) Declare a structure **time** with following members: [3]

hh hour (integer type)
mm minute (integer type)

Declare another structure **multiplex** with following members:

scrno screen number (integer type)
movie movie name (string type)
start starting time (time type)

Create a variable of the type **multiplex** in the **main**() function. Use an initializer to assign following values to the variable: screen number is **3**, movies name is **DANGAL** and starting time **6:30** PM. Display the values stored in the variable on the screen.

```
struct time { int hh, mm; };
```

```
struct multiplex { int scrno; char movie[25]; time start; };
void main()
{
   multiplex m={3, "DANGAL", {6, 30}};
   cout<<"Screen Number="<<m.scrno<<endl;
   cout<<"Movie="<<m.movie<<endl;
   cout<<"Starts At="<<m.start.hh<<":"<< m.start.mm<<endl;
}
```

b) Declare a structure **sportsclub** with the following members: [5]

memid member ID (integer type)
mname member name (string type)
sports sporting activity (string type)
fees monthly fees (double type)

Monthly Fees is charged according to the table given below:

Sports Activity	CRICKET	FOOTBALL	BASKETBALL	VOLLEYBALL
Monthly Fees	4500	3000	2000	1500

Write a C++ function **scinput**(), to input values for **memid**, **mname** and **sports** for a parameter of the type **sportsclub** and calculate **fees**. Write another C++ function **scshow**() to display values stored in the parameter of the type **sportsclub**. In the **main**() function create a variable of the type **sportsclub**; call the functions **scinput**() and **scshow**().

```
struct sportsclub
{
   int memid; char memname[25], sports[15]; double fees;
};
void scinput(sportsclub &a)
{
   cout<<"Member ID? "; cin>>a.memid;
   cout<<"Member Name? "; gets(a.memname);
   cout<<"Sports? "; gets(a.sports);
   if (strcmpi(a.sports, "CRICKET")==0)
      a.fees=4500;
   else
   if (strcmpi(a.sports, "FOOTBALL")==0)
      a.fees=3000;
   else
   if (strcmpi(a.sports, "BASKETBALL")==0)
      a.fees=2000;
   else
      a.fees=1500;
}
void scshow(sportsclub b)
{
   cout<<"Member ID="<<b.memid<<endl;
   cout<<"Member Name="<<b.memname<<endl;
   cout<<"Sports="<<b.sports<<endl;
   cout<<"Fees="<<b.fees<<endl;
}
void main()
{
   sportsclub sc;
   scinput(sc);
   scshow(sc);
}
```

5. a) Write a C++ function to sort an array of **double** using **selection** sort. Array name and number of elements in the array are passed as parameters to the function. [3]

```
void selectionsort(double a[], int n)
{
   for (int k=0; k<n-1; k++)
   {
      double t=a[k]; int p=k;
      for (int x=k+1; x<n; x++)
         if (a[x]<t)
         {
            t=a[x];
            p=x;
         }
      a[p]=a[k];
      a[k]=t;
   }
}
```

b) Suppose **ar1**, **ar2** and **ar3** are three arrays of integers with **n1**, **n2**, and **n1**+**n2** elements respectively. Merge arrays **ar1** and **ar2** to obtain the third array **ar3**. Array **ar1** is sorted in descending order, array **ar2** is sorted in ascending order and array **ar3** to appear in ascending order. Use the function declaration given below for merging:

```
void merge(int ar1[], int ar2[], int ar3[], int n1, int n2);   [4]
void merge(int ar1[], int ar2[], int ar3[], int n1, int n2)
{
   int i=0, j=n2-1, k=0;
   while (i<n1 && j>=0)
      ar1[i]<ar2[j] ? ar3[k++]=ar1[i++] : ar3[k++]=ar2[j--];
   while (i<n1)
      ar3[k++]=ar1[i++];
   while (j>=0)
      ar3[k++]=ar2[j--];
}
```

6.
```
struct employee
{
   int code; char name[20]; double itax;
};
```

a) Write a C++ function to sort an array of **employee** using **insertion** sort on **itax** in descending order. Array name and number of elements in the array are passed as parameters to the function. [3]

```
void insertionsort(employee a[], int n)
{
   for (int k=1; k<n; k++)
   {
      employee t=a[k];
      int x=k-1;
      while (x>=0 && t.itax>a[x].itax)
      {
         a[x+1]=a[x];
         x++;
      }
      a[x+1]=t;
   }
}
```

b) Write a C++ function to sort an array of **employee** using **bubble** sort on **name**. Array name and number of elements in the array are passed as parameters to the function. [3]

```
void bubblesort(employee a[], int n)
{
   for (int k=1; k<n; k++)
      for (int x=0; x<n-k; x++)
         if (strcmpi(a[x].name, a[x+1].name)>0)
         {
            employee t=a[x];
            a[x]=a[x+1];
            a[x+1]=t;
         }
}
```

c) Write a C++ function to binary search for a **code** in a sorted array of **employee** sorted on **code**. Array name, number of elements in the array and the **code** to be searched are passed as parameters to the function. If **code** is located in the array then display **code**, **name** and **itax**, otherwise display an error message "**Code Not Found in the Array**". [4]

```
void binarysearch(employee a[], int n, int code)
{
   int lb=0, ub=n-1, found=0, mid;
   while (lb<=ub && found==0)
   {
      mid=(lb+ub)/2;
      if (code<a[mid].code)
         ub=mid-1;
      else
      if (code>a[mid].code)
         lb=mid+1;
      else
         found=1;
   }
   if (found==1)
      cout<<a[mid].code<<","
          <<a[mid].name<<","
          <<a[mid].itax<<endl;
   else
      cout<<"Code Not Found in the Array\n";
}
```