1.Ans. (*a*) Procedural Programming :
(*i*) The emphasis is on doing things rather than data.
(*ii*) It is based on the sequence of instructions.

Object Oriented Programming : (1.5 Marks for each)
(*i*) It emphasises on the data.
(*ii*) It is based on the principle of data hiding, abstraction, inheritance and polymorphism.

(*b*) (*i*) ctype.h (*ii*) stdio.h (*iii*) stdlib.h (*iv*) string.h (1/2 Marks for each correction)

(*c*) The correct program is :

```
#include <iostream.h> //                Correction - 1  (1/2 Marks for each correction)
void main()
{
int X,Y;
cin>>X;
for (Y=0; Y<10; Y++) //                 Correction - 2
If ( X == Y ) //                        Correction - 3
cout<<Y+X;
else
cout<< Y; //                            Correction - 4
}
```

Correction 1 : () bracket cannot be used with #include
Correction 2 : In place of , symbol ; should be used and y should be Y
Correction 3 : () bracket was missing with if
Correction 4 : '<<' should be used with cout

(*d*) The output is : cOmmUTer (4 Marks)

(*e*) The output is as : (4 Marks)

```
190 280 100
280 10
290 570 10
570 10
```

(f) The output is : 10,30,20 (3 Marks)

2. Ans. (*a*) A constructor is a special initialization function that is called automatically whenever an instance of your class is declared. The name of the constructor is same as that of class and it returns no value.This function is the opposite of the constructor in the sense that it is invoked when an object ceases to exit. It also has a same name as that of class but with a prefix '~'. The difference between the constructor, destructor and other functions is that the functions can return a value but the constructor and destructor do not return any value. (3 Marks)

(*b*) // Class and function declaration of employee class (5 Marks)

```
class employee
{
int empno;
char ename[20];
float basic, da, hra;
float netpay;
float calculate()
{
return (basic + da + hra);
}
public :
void havedata()
{
cout << "Enter employee no. : ";
cin >> empno;
cout << "Enter name : ";
cin >> ename;
cout << "Enter basic salary : ";
cin >> basic;
cout << "Enter DA : ";
cin >> da;
cout << "Enter HRA : ";
cin >> hra;
netpay = calculate();
}
void dispdata()
{
cout << "Employee no is : " << empno<<endl;
cout << "Name is : " << ename << endl;
cout << "Basic Salary : " << basic<<endl;
cout << "DA is : " << da<<endl;
cout << "HRA is : " << hra << endl;
cout << "Netpay is : " << netpay<<endl;
} };
```

(c) The class is as : (4 Marks)

```
class COMPETITION
{
private :
int event_no;
char description[30];
int score;
char qualified;
public :
void input()
{
cout << "Enter event no. : ";
cin>>event_no;
cout << "Enter description : ";
gets(description);
cout << "Enter score : ";
cin>>score;
cout << "Enter qualified : ";
cin>>qualified;
}
void Award(int cut_off)
{
if (score > cut_off)
qualified = 'y';
else
qualified = 'N';
}
void show()
{
cout << event_no << description << score
<< qualified;
}
};
```

3.Ans. (*a*) // This function search an element in an array using binary search. (5 Marks)

```
int binary(float ARR[10], float data ,int n)
{
int i, flag = 0, first = 0, last, pos = 1, mid;
last = n - 1;
while ((first <= last) && (flag == 0))
{
mid = (first + last) / 2;
if (ARR[mid] == data)
{
pos = pos + mid;
flag = 1;
}
else
if (ARR[mid] < data)
first = mid + 1;
else
last = mid - 1;
}
if (flag == 1)
return(1);
else
return(0);
}
```

(*b*) Let us assume that the Base index number is [0][0]. [3+3]

Number of Rows = R = 10
Number of Columns = C = 15
Size of data = W = 4
Base address = B = S[0][0] = 1000
Location of S[8][9] = X

(*i*) When S is stored by Row Major

X = B + W * [8 * C + 9]
= 1000 + 4 * [8 * 15 + 9]
= 1000 + 4 * [120 + 9]
= 1000 + 4 * 129
= 1516

(*ii*) When S is stored by Column Major

X = B + W * [8 + 9 * R]
= 1000 + 4 * [8 + 9 * 10]
= 1000 + 4 * [8 + 90]
= 1000 + 4 * 98
= 1000 + 392
= 1392

(*c*) // Function to find the sum of row elements of a two dimensional array [4]

```
void calculate(int R[5][6])
{
int i, j, sum;
sum=0;
for(i=0; i<5; i++)
{
sum=0;
for(j=0; j<6; J++)
{
sum = sum + R[i][j];
}
cout << "\n\t sum of " << (i+1) << " row is " << sum;
}
}
```

(*d*) The stack operation is : [4]

Scanned elements	Operation	Stack
100	PUSH 100	100
40	PUSH 40	100, 40
8	PUSH 8	100, 40, 8
+	POP 8	
	POP 40	
	Calculate 40 + 8 = 48	
	PUSH 48	100, 48
20	PUSH 20	100, 48, 20
10	PUSH 10	100, 48, 20, 10
−	POP 10	
	POP 20	
	Calculate 20 − 10 = 10	
	PUSH 10	100, 48, 10
+	POP 10	
	POP 48	
	Calculate 48 + 10 = 58	
	PUSH 58	100, 58
*	POP 58	
	POP 100	
	Calculate 58 * 100 = 5800	
	PUSH 5800	5800

(*e*) SELECT name. charges. age FROM HOSPITAL WHERE SEX = "M";
(*f*) SELECT COUNT(*) FROM HOSPITAL WHERE age > 20:
(g) INSERT INTO HOSPITAL VALUES (11, "Mustafa", 37, "ENT", {25/02/98}, 250, "M");
(*h*) (*i*) 5 (*ii*) 12 (*iii*) 1600 (*iv*) 387.50 (1/2 marks each)