*Choose the correct option:-

1- D
2- C
3- A
4- C
5- True

*Short answer type Question:-

1- Delete is an operator that is used to destroy array and non-array(pointer) object which are created by new expression .
New operator is used for dynamic memory allocation which puts variable on heap memory .

```
 #include <iostream>
using namespace std;

int main()
{
   int *p1, *p2, sum;
   p1=new int;
   p2= new int;
   cout<<" Enter first value :";
   cin>>*p1;
    cout<<" Enter second value :";
   cin>>*p2;
   sum = *p1 + *p2;
   cout<<" Sum of values = "<<sum<<endl;
   delete p1;
   delete p2;
}
```

OUTPUT

```
 Enter first value :5
 Enter second value :5
 Sum of values = 10
```

2- A constructor is a special type of function with no return type. Name of constructor should be same as the name of class.

Constructor types:-
1- Default Constructor
2- Parameterized Constructor
3- Copy Constructor
4- Static Constructor
5- Private Constructor

3- POP
i- Program is divoded in small parts called function.
ii-Procudure Oriented Programming follow pop down approach.
iii-These is no access specifies in procedure programming.
iv- Adding new delete function is not easy.

OOP
i- Program is divided in small part called object.
ii- Object Oriented program follows bottom up approach.
iii-OOP has access specific like private, public, protected etc.
iv- Adding new delete & function is easy.

Long Answer Type Question:-
4- Polymorphism is mainly divided into two type:-
i- Compile Time Polymorphism
ii- Run Time Polymorphism

i- CTP- This type of polymorphism is achived by function overloading or operator overloading when there are multiple functions with same name but different parameters then these function are said to be overloaded .

Ex-

```
Class Complex{
Private:
int read, image;
Public:
Complex( int r = 0, int i = 0 ) {read= r, image = i;}
Complex operator + (Complex const & obj)
{
Complex res;
res. read = read + obj. read;
res. img = image + obj. image;
return res.;
}
void print()
{
cout<<read<<"+i"<<image<<read<<endl;
};
}
```

ii- RTP - This type of polymorphism is achived by function overriding. When a derived class has a defination for one of the member functions of the base class . That base function is said to be overridding .

Ex-

```
Class base {
public:
Virtual void print () {
cout<<"Print base class"<<endl;}
void show () {
cout<<"Show base class"<<endl;}
};
int main()
{
 base * bpts;
 derived d;
 bptr = &d;
 bptr -> print();
 bptr -> show();
 return 0;
 };
 }

5-
 void sort012(int a[], int arr_size)
{
      int lo = 0;
   int hi = arr_size - 1;
   int mid = 0;
   while (mid <= hi) {
   switch (a[mid]) {
       case 0:
           swap(a[lo++], a[mid++]);
           break;
       case 1:
           mid++;
           break;
       case 2:
           swap(a[mid], a[hi--]);
           break;
       }
   }
}
void printArray(int arr[], int arr_size)
{
   for (int i = 0; i < arr_size; i++)
       cout << arr[i] << " ";
}
```

```
int main()
{
    int arr[] = { 0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1 };
    int n = sizeof(arr) / sizeof(arr[0]);

    sort012(arr, n);

    cout << "array after segregation ";

    printArray(arr, n);

    return 0;
}

 OUTPUT - 0 0 0 0 0 1 1 1 1 1 2 2

 6 -

using namespace std;

class member{
   char name[20], address[40];
   double number;
   int age;
   public:
    int salary;
    void input()
    {   cout<<endl;
     cout<<"Name : "<<endl;
     cin.getline(name, 20);
     cout<<"Age : "<<endl;
     cin>>age;
 cout<<"Phone Number : "<<endl;
     cin>>number;
     cout<<"Address : "<<endl;
     cin.getline(address, 40);
     cout<<"Salary : "<<endl;
     cin>>salary;

 }
 void display()
    {   cout<<endl;
     cout<<"Name : "<<name<<endl;
     cout<<"Age : "<<age<<endl;
     cout<<"Phone Number : "<<number<<endl;
     cout<<"Address : "<<address<<endl;
     cout<<"Salary : "<<salary<<endl;

 }
};
  class manager : public member{
  void printSalary()
          {
         cout<<"\n Salary of the member is : "<<salary<<endl;
  }
};
int main()
{
 m. display();
 m. print.salary();
 }
```