1) Local variables are stored in an area called

Ans) Option (d) → Stack

2) #include using namespace std;

```
Class Base { };

Class Derived : Public Base { };

int main()
{
  Base *bp = new Derived;
  Derived *dp = new Base;
}
```

Ans:- option (c)
→ Compiler Error in line "Derived *dp = new Base;".

3) When the inheritance is private, the private methods in base class are _____ in the derived class (in C++)

Ans:- option (a) → Inaccessible

Date:
Page:

4) which of the following is true?

Ans:- Option (a) → The Number of times destructor is called depends on Number of objects created.

5) State True or False :
Type Conversion is automatic whereas type Casting is explicit

Ans:- Option (A) → True

Short Answer type questions?

1) Explain about new and delete keywords with code.

new operator :- This keyword is actually unary operator which is used to allocate space dynamically or we can say this keyword is used for allocating the memory dynamically that at is at the Runtime.

delete keyword:- when the new keyword allocates memory dynamically, it is only required for specific period of time within a program and when it is no longer required it should be

freed, so that memory would be available again for other demands of dynamic memory. So for this purpose 'delete' keyword or operator is used.

⇒ Syntax for new operator :

~~new~~ int

```
int *p = new int;
```

The above line will dynamically allocate memory to contain one single element of type int and store the address in p.

For arrays :-

Syntax ;

~~int *val = NULL; // pointer initialized with NULL value~~

```
int *val = new int[10];
```

For delete operator :-

Simply

delete pointer name

example:

```
delete p;
delete []p; // for array of elements.
```

Date:
Page:

Code for example:

```
#include <bits/stdc++.h>
  using namespace std;
int main()
{
   int *p;
    p= new int; //dynamic allocation
*p= 45;
   cout << *p << endl;    //45 printed

delete p;    //deallocated memory

}
```

2) What are constructors? why they are required? Explain different types of constructors with suitable example.

Ans:- A constructor is a member function of a class which initializes objects of a class. and it is invoked automatically at time of object creation. Constructors are required:
To construct an object of the class, in other words, it is used to initialize all class data members, if we don't write a constructor, compiler will provide default constructor in C++ programming.

The different types of constructors are:-

(i) Default constructor:-

Default constructor is the constructor which doesn't take any argument, It has no parameters, and it is ~~call~~ provided by the compiler also in the case we didn't defined a Constructor

example:-

```
#include <bits/stdc++.h>
using namespace std;
class Default
{
  Public:
  int a, b;

Default(){ ~~construct ()~~ // Default Constructor
  {
    a = 1;
    b = 2;
  }
};
int main() {
construct c; // Default constructor
  called automatically on object creation
```

Date:
Page:

```
cout<< a <<" " << b <<endl;
 return 1;
}
```

output

1 2

(ii) Parameterized Constructor:-

A constructor in which it is possible to pass arguments. This type of constructor helps initializing an object when it is created.

example:

```
#include <bits/stdc++.h>
 using namespace std;

class Parameter
{
  Private:
   int a, b;
 Public:
    Parameter ( int a1, int b1 )
    {
       a = a1;
       b = b1;
    }
 int getA()
    {
      return a;
    }
```

```
int getB()
  {
    return b;
  }
};
int main()
{
  Parameter P(10,15); //constructor called
cout << "A = " << p.getA() << ", B = "
   << p.getB() << endl;
   return 0;
}
```

output :-

A = 10, B = 15

(iii) Copy constructor:- A copy constructor is a member function which initializes an object using another object of the same class;

Example:

```
#include <bits/stdc++.h>
using namespace std;
class Copy
{
  private:
    double X, Y;
```

```
Public:
    copy(double x1, double y1)
    {
        x = x1;
        y = y1;
    }
};
int main(void)
{
    copy c = copy(5,6);
}
```

3) Explain the difference b/w Object Oriented and procedural programming language in detail.

	POP (procedural)	OOP (object oriented...)
(i)	It is process-oriented	It is Object (result) - oriented.
(ii)	It follows top-down approach	It follows Bottom-Up approach
(iii)	Each function contains different data	Each object controls its data.
(iv)	Overloading is not possible	Overloading is possible

(v)	it is less secure	it is more secure
(vi)	Addition of new data and function is not easy	Adding new data and functions is easy.

The Procedural Programming is a Structured programming approach, based on concept of calling procedures which contains series of steps to be computed.

Object-Oriented is based upon the concept of Objects and Classes.
It focuses on the data rather than the algorithm to create the modules by dividing into the data and functions.

Long answer type question

A) Explain the type of Polymorphism with code.

Ans:- Polymorphism is a crucial feature of OOP and it can be defined as the ability of a message to be displayed in more than one form basically 'One name, many forms'.

There are two types of Polymorphism
(i) Compile Time Polymorphism
(ii) Run Time Polymorphism

(i) Compile time Polymorphism :- This type of Polymorphism is achieved by function overloading

⇒ Function Overloading :- when there are multiple functions with same name but different parameters then the functions are said to be overloaded.

By change in number of arguments or change in type of arguments we can achieve Overloading.

example:-

```
#include <bits/stdc++.h>
Using namespace Std;
Class Overload {

Void func (int); //assuming implementation
Void func (double); //of these overloaded
                    //functions
};

int main ()
{
    Overload obj;
    obj.func (7);
    obj.func (8.412);
}
```

Operator Overloading:- we can also Overload operators in C++, for example '+' can be used for addition of int types or for concatenating two strings, so basically operator Overloading is under Compile time Polymorphism.

Date:
Page:

Example:

```
#include <bits/stdc++.h>
using namespace std;
class Test
{
  private :
    int num;
  public :
    Test () : num(8)
    {
    }
    void operator ++ ()
    {
      num = num+2;
    }
    void display ()
    {
      cout << "The Count =" << num;
    }
};
int main()
{
  Test t;
  ++t;    // calling of "void operator ++()"
  t.display();
  return 0;
}
```

Output : The Count = 10

Date:
Page:

Runtime polymorphism :-
This is achieved by Overriding function
This function Overriding occurs when a derived class has a definition for one of the member functions of the base class. That base function is Overriden.

Runtime Polymorphism occurs when there is a hierarchy of classes and they are related by inheritance.

```
#include <bits/stdc++.h>
using namespace std;
class base
{
  public:
     void print ()
     {
      cout << " print base class" <<endl;
     }
    void show ()
    {
      cout << " show base class"<<endl;
    }
};

class derived : public base
  {
    public :
```

Date:
Page:

```
void print()
  {
    cout << "print derived class" << endl;
  }
  void show()
  {
     cout << "show derived class" << endl;
  }
};

int main()
{
    base *b;
    derived d;
      b = &d;
    b -> print();
    b -> show();
    return 0;
}
```

Output:

print base class
show base class.

6) Write a program to sort an array of 0, 1, 2 in the best possible time and space complexity

Input array: 1 1 2 2 0 0 2 1 2
Output array: 0 0 1 1 1 2 2 2 2

Sol:

```
#include<bits/stdc++.h>
Using namespace Std;

void Sort(int arr[], int n)
{
  int low = 0, mid = 0, high = n-1;
   while (mid <= high)
   {
      if (arr[mid] == 0)
       {
          Swap(arr[mid], arr[low]); //in-built
//function in the C++ (STL) which swaps two
// value
          low = low+1;
          mid = mid+1;
       }

      else if (arr[mid] == 2)
       {
         swap(arr[mid], arr[high]); // Same in-built
          high = high -1;
       }
```

```
        else
        {
          mid = mid + 1;
        }
    }
}

int main()
{
    int n, i;
    cout << "Enter Size of the array" << endl;
    cin >> n;
    int arr[n];

    for(i=0; i<n; i++)
    {
        cin >> arr[i]; // Input (for array elements)
    }

    Cout << "Array before Sorting" << endl;
    for(i=0; i<n; i++)
    {
        cout << arr[i] << endl;
    }

    cout << endl;  // just for next line for displaying
                   // in output
```

```
cout << "Sorted array is" << endl;
    sort (arr, n); // calling the sort
                   // function
for(i=0; i<n; i++)
  {
    cout << arr[i] << endl;
  }
}
```

c) Create a class named 'Member' having the following members:

Data members

1- Name
2- Age
3- Phone number
4- Address
5- Salary

It also has a method named 'printSalary' which prints the salary of the members.

Two classes 'Employee' and 'Manager' inherits the 'Member' class. The 'Employee' and 'Manager' classes have data members 'specialization' and 'department' respectively. Now

Date:
Page:

assign name, age, phone number, address and salary to an employee and a manager by making an object of both of these classes and print the same.

```
#include <bits/stdc++.h>
using namespace std;

class Member
{ public:
   char name[20], address[30];
   long phn_num;
   int age;
   double salary;

  void Enter_details()   //for input of details
   {
    cout<<endl;  // for new line
    cout<<"Name :"<<endl;
    cin.getline(name,20); // for space separated name

    cout<<"Age :"<<endl;
    cin>>age;
    cout<<"Phone Number :"<<endl;
    cin>>phn_num;
    cout<<"Address :"<<endl;
    cin.getline(address,30);
    cout<<"Salary :"<<endl;
    cin>>Salary;
   }
```

Date:
Page:

```
void Print_Salary()
{
    cout << " The Salary of the Member is :" << salary
        << endl;
}
```

```
Void display ()
 {
  cout<<endl;
  Cout<< "Name : " <<name <<endl;
  Cout<< "Age : " << age<<endl;
  Cout<< " Phone Number" <<phn_num<<endl;
  cout<<"Address : " <<address<< endl;
  ~~cout<< " Salary : " <<salary << endl;~~
 }
};

class Employee : public Member
 {
  char spec[20] , dept[20];
  public :

  Void input ()
  {
   cout << " Enter Employee Details \n";
   Member :: Enter_details ();
   Cout<< " Specialization : " <<endl;
   Cin.getline (Spec, 20);
   cout<< " Department : " <<endl;
   Cin.getline (dept, 20);
  }
```

```
void display()
 {
   cout<<"\n Displaying Employee Details \n";
    Member::display();
   cout<<"Specialization:"<<spec<<endl;
   cout<<"Department:"<<dept<<endl;
        print_Salary();   // inside display().
   }
   ~~print~~"
};

class Manager : public Member
 {
  char spec[20], dept[20];
  Public:
   void input()
   {
   cout<<"\n Enter Manager details \n";
   Member::Enter_details();
   cout<<"Specialization :"<<endl;
   cin.getline(spec, 20);
  cout<<"Department :"<<endl;
   cin.getline(dept, 20);
   }

  void display()
  {
  cout<<"\n Displaying Manager details \n";
  Member::display();
```

```
cout<< "Specialization :" <<spec <<endl;
cout<<"Department :"<<dept <<endl;
    }
    Print_Salary();          }
                             };

int main()
{
    Employee e;
    Manager m;

    e.Enter_details();
    m.Enter_details();
    e.display();
    e.print. m.display();
    }
```