PRACTICAL-1:

PROGRAM TO DEMONSTRATE USE OF DATA MEMBERS & MEMBER FUNCTIONS.

```
#include <iostream.h>
#include <string.h>  using
namespace std;  #define
MAX_CHAR 30   class
person
{
        private:
          char name  [MAX_CHAR];
          int age;
public:
        void get(char n[ ], int a)
        {
          strcpy(name , n);
age = a;
}
void put()
{
cout<< "Name: " <<name<<endl;
```

```
cout<< "Age: " <<age <<endl;
} };
int main()
{
clrscr();
class person PER;     PER.get("Manisha", 32);
PER.put();   return
0;
}
```

PRACTICAL-2A:

PROGRAMS BASED ON BRANCHING AND LOOPING STATEMENTS USING CLASSES.

```
#include <iostream.h>
#include <conio.h>
class voter
{
public: int age:
void display()
{
cout<<"Enter Your Age: "<<endl;
cin>>age;
if (age›=18)
        cout<<'Your Are Eligible For voting"<<<endl;
        }
}:
        void main()  {
clrscr();          voter
v;
       v.display();
getch);
        }
```

PRACTICAL-2B:

```
#include<iostream.h>
#include <conio.h> class
EvenOdd
{
public: int no ; void
display()
{
cout <<"Enter A Number: "; cin>>no;
if(no%2==0) cout<<"Given Number is
Even"<<endl; else
cout<<"Given Number is Odd'<<endl;
}  };
void main()
{ clrscr();
EvenOdd ed;
ed .display();
getch();
}
```

PRACTICAL-2C:

```
#include<iostream.h>
#include <conio.h>  void
main()
{ int i,j; clrscr(); for
(i=1 ; i<=5 ; i++)
{ for (j=1 ; j<=i ;
j++)
{
cout<<"*";
}
cout <<endl;
}
 getch();
}
```

PRACTICAL-2D:

```
# include<iostream.h>
#include <conio.h> void
main()
{
int num , res=0 , remainder ; clrscr();
cout<<"Enter Number: "<<endl ;
cin>>num; while (num! =0)
{
remainder=num%10     ;
res=res*10+remainder ;
num=num/10 ;
cout<<"Reverse Number is: "<<endl<<res ;  getch();
}
```

PRACTICAL-3A

C++ PROGRAM FOR ONE DIMENSIONAL ARRAY.

```
#include<iostream.h>
#include <conio.h>
void main()
{
clrscr();
int arr[50] , num , i ;
cout<<"\n How Many Elements You Want to Store into an Array? ";
cin>>num;
cout<<"\n Enter "<<num<<"Elements to Store into an Array : ";
for(i=0 ; i<num ; i++)
{
cin>>arr[i];
}
cout<<"\n The Elements in the Array are : ";
for(i=0 ; i<num ; i++)
{
cout<<arr[i]<<"\t";
}
 getch();
}
```

PRACTICAL-3B

C++ PROGRAM FOR TWO DIMENSIONAL ARRAY.

```
#include<iostream.h>
#include <conio.h>
Void main()  {
      int arr[50][50], m, n, i, j ;
cout<<" How Many Rows and Columns You Want to Store into an Array? ";
cin>>m>>n;
cout<<"\n Enter Elements into Array : \n";
      for(i=0 ; i<m ; i++)
{
for(j=0 ; j<n ; j++)
      {
            cin>>arr[i][j];
      }
}
      cout<<"\n The Elements in the Array are : \n";
      for(i=0 ; i<m ; i++)
{
for(j=0 ; j<n ; j++)
      {
```

```
            cout<<arr[i][j]<<"\t";
        }
        getch();
}
```

PRACTICAL-4

```
#include <iostream>
#include<conio.h>
using namespace std;
char a = 'm';
static int b = 50;
int main()
{
char a = 's';
cout << "The static variable : "<< ::b;
cout << "\nThe local variable : " << a;
cout << "\nThe global variable : " << ::a;
return 0;
}
```

PRACTICAL-5A

DEFAULT CONSTRUCTOR

```
#include <iostream.h>
#include<conio.h>
 class pract5a
 {
public: int y, z;
pract5a()
{
y = 7;
z = 13;
}
};
void main()
{
pract5a p;
cout <<"the sum is: "<< p.y+p.z;
getch();
}
```

PRACTICAL-5B

PARAMETERIZED CONSTRUCTOR

```
#include <iostream.h>
#include<conio.h>
class pract5b
{
public: int x;
pract5b(int x1)
{
x = x1;
}
int getX()
{
return x;
}
};
void main()
{
pract5b p(10);
cout << "p.x = " << p.getX() ;    getch();
}
```

PRACTICAL-6

C++ PROGRAM TO DEMONSTRATE PUBLIC ACCESS MODIFIER

```
#include <iostream.h>
#include<conio.h>
class Circle
{
  public:
 double radius;
 double area()
       {
 return 3.14*radius*radius;
       }
};
void main()
{
Circle C;
C.radius = 5.5;
cout << "Radius is: " << C.radius
cout<<Area<<A.area();
getch();
}
```

PRACTICAL-7A

SINGLE INHERITANCE

```
#include <iostream.h>
#include<conio.h>
class base
 {
public: int x;
void getdata()
{
cout << "Enter the value of x = "; cin >> x;
}
};
class derive : public base
 {
private: int y;
public: void readdata()
{
cout << "Enter the value of y = "; cin >> y;
}
void product()
{
cout << "Product = " << x * y;
```

```
}
};
void  main()
{
clrscr();
derive d;
d.getdata();
d.readdata();
d.product();
getch();
}
```

PRACTICAL-7B

MULTILEVEL INHERITANCE

```
#include <iostream.h>
#include<conio.h>
class base  {
public:
int x;
void getdata()    {
cout << "Enter value of x= ";
cin >> x;
}  };
class derive1 : public base
{
public:
int y;
void readdata()
{
cout << "\nEnter value of y= "; cin >> y;
}  };
class derive2 : public derive1
{
private:
```

```
int z;
public:
void indata()
{
cout << "\nEnter value of z= "; cin >> z;
}
void product()
{
cout << "Product= " << x * y * z;
}
};
void main()
{
clrscr ();
derive2 a;
 a.getdata();
a.readdata();
a.indata();
a.product();
getch();
}
```

PRACTICAL-8A

MULTIPLE INHERITANCE

```
#include<iostream.h>
#include <conio.h>
class A
{
public:
int x;
void getx()
{
cout << "enter value of x: ";
cin >> x;
}
};
class B
{
public:
int y;
void gety()
{
cout << "enter value of y: ";
cin >> y;
```

```
}
};
class C : public A, public B
 {
public:
void sum()
{
cout << "Sum = " << x + y;
}
};
 void main()
{
C c1;
c1.getx();
c1.gety();
c1.sum();
 getch();
}
```

PRACTICAL-8B

HIERARCHICAL INHERITANCE

```
#include<iostream.h>
#include <conio. h>
class A
{
public:
int x, y;
void getdata()
{
cout <<"Enter value of x  y:\n";
cin >> x >> y;
}
};
class B : public A
{
public:
void product()
{
cout << "\nProduct= " << x * y;
}
};
```

```
class C : public A
{
public:
void sum()
{
cout << "\nSum= " << x + y;
}
};
void main()
{
B b1;
C c1;
b1.getdata();
b1.product();
c1.getdata();
c1.sum();
getch();
}
```

PRACTICAL-9

```
#include<iostream.h>
#include <conio.h>
class A
{
public:
A()
{
cout << "Constructor of the base class A \n";
}
};
class B
{
public:
B()
{
cout << "Constructor of the base class B \n";
}
};
class C: public A, virtual B
{
public:
```

```
C(): A(), B()
{
cout << "Constructor of the derived class C \n";
}
};
void main()
{
clrscr();
C obj;
getch();
}
```

PRACTICAL-10A

C++ PROGRAM TO DEMONSTRATE FRIEND FUNCTION.

```
#include <iostream.h>
#include <conio.h>
class Distance
{
private:
int meter;
public:
Distance(): meter(0)
{
}
friend int func(Distance);
};
int func(Distance d)
{
d.meter=10;
return d.meter;
}
void main()
{
clrscr();
```

```
Distance D;
cout<<"Distace: "<<func(D);
getch();
}
```

PRACTICAL-10B

INLINE FUNCTION

```
#include <iostream.h>
#include <conio.h>
inline int Max(int x, int y)
{
return (x > y)? x : y;
}
void main()
{
cout << "Max (20,10): " << Max(20,10) << endl;
cout << "Max (0,200): " << Max(0,200) << endl;
cout << "Max (100,1010): " << Max(100,1010) << endl;
getch();
}
```

PRACTICAL-10C

THIS FUNCTION

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
class Student
{
public:
char Name[50];
float ID;
Student(char* Name, float ID)
{
strcpy(this -> Name, Name);
this -> ID = ID;
}
void details()
{
cout << "Student Name: " << Name << endl;
cout << "Student ID: " << ID << endl << endl;
}
};
```

```
void main()
{
clrscr();
Student s1("Neel", 1);
Student s2("Yashaswi", 2);
s1.details();
s2.details();
getch();
}
```

PRACTICAL-11A

PROGRAM OF FUNCTION OVERLOADING WHEN NUMBER OF ARGUMENTS ARE DIFFERENT

```
#include <iostream.h>
#include <conio.h>
class Cal {
public:
static int add(int a,int b)
{
return a + b;
}
static int add(int a, int b, int c)     {
return a + b + c;
}
};
void main ()
{
clrscr ();
Calc C;
cout<<C.add(10, 20)<<endl;
cout<<C.add(12, 20, 23);
getch();    }
```

PRACTICAL-11B

PROGRAM OF FUNCTION OVERLOADING WHEN DATA TYPE OF ARGUMENTS ARE DIFFERENT

```
#include <iostream.h>
#include <conio.h>
int plus(int x, int y)
{
return x + y;
}
double plus(double x, double y) {
return x + y;
}
void main ()
{
clrscr();
int Num1 = plus(8, 5);
double Num2 = plus(4.3, 6.26);
cout << "Int: " << Num1 << "\n";
cout << "Double: " << Num2;
getch();
}
```