1. Write a C++ program to sort the elements in ascending and descending order.

```
#include <iostream.h>
#include<conio.h>
int main()
{
        int num[100] , n;
        int i, j, man;
        clrscr();
       cout<<"\n Enter the size of an array"<<endl;
       cin>>n;
       cout<<"\n Enter values for the array elements"<<endl;
       for( i=0; i<n; i++ )
            {

                cin>>num[i];
           }
       For (i=0;i<n;i++)
            {
                for(j=0;j<n;j++)
                    {
                        if(num[i]<num[j])
                            {
                            man=num[i];
                            num[i]=num[j];
                            num[j]=man;
                          }
                    }
            }
        cout<<"\n Elements in ascending order "<<endl;
        for (i=0; <n; i++)
              {
               cout<<"\t"<<num[i]<<endl;
             }
        for(i=0;i<n;i++)
              {
               for(j=0;j<n;j++)
                        {
                         if(num[i]>num[j])
                               {
                            man=num[i];
                            num[i]=num[j];
                            num[j]=man;
                          }
                    }
              }
        cout<<" \n Elements in descending order"<<endl;
        for(i=0;i<n;i++)
           {
               cout<<"\t "<<num[i]<<endl;
           }
    return 0;
}
```

Output:

Enter the size of an array

5

Enter values for the array elements

9 2 5 1 0

Elements in ascending order

0 1 2 5 9

Elements in descending order

9 5 2 1 0

2. Write a C++ program to find the sum of all the natural numbers from 1 to n.

```
#include <iostream>
using namespace std;
int main()
{
   int n, sum = 0;
   cout << "Enter a positive integer: ";
   cin >> n;
   for (int i = 1; i <= n; ++i) {
      sum += i;
   }
   cout << "Sum = " << sum;
   return 0;
}
```

Output:

Enter a positive integer: 9

Sum = 45

3. Write a C++ program to swap 2 values by writing a function that uses call by reference technique.

```
#include<iostream>
using namespace std;
void swap(int &x, int &y)
{ int swap;
 swap=x;
 x=y;
 y=swap;
}
```

```
int main()
{     int x=500, y=100;
 swap(x, y);  // pass by reference
 cout<<"Value of x is: "<<x<<endl;
 cout<<"Value of y is: "<<y<<endl;
 return 0;
}
```

Output

Value of x is: 100

Value of y is: 500

4. Write a C++ program to demonstrate function overloading for the following prototypes.

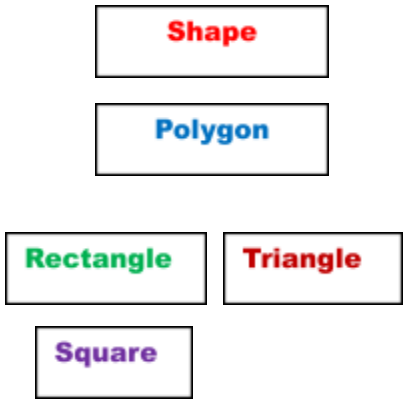
```
// overload.cpp
void add(int a, int b);
void add(double a, double b);
#include <iostream>
using namespace std;
 void add(int a, int b)
{
  cout << " \n sum = " << (a + b);
}
void add(double a, double b)
{
   cout << " \n sum = " << (a + b);
}
```

```
int main()
{
    add(10, 2);
    add(5.3, 6.2);
     return 0;
}
Output:
sum = 12
sum = 11.5
```

5. Create a class named Shape with a function that prints "This is a shape". Create another class named Polygon inheriting the Shape class with the same function that prints "Polygon is a shape". Create two other classes named Rectangle and Triangle having the same function which prints "Rectangle is a polygon" and "Triangle is a polygon" respectively. Again, make another class named Square having the same function which prints "Square is a rectangle". Now, try calling the function by the object of each of these classes.

```
// shape.cpp
#include <iostream>
using namespace std;
class Shape
{
   public:
   Shape(){}
   void print(){
      cout<<"\nThis is a shape.";
   }
};
class Polygon: public Shape
{
   public:
   Polygon(){}
   void print(){
      cout<<"\nPolygon is a shape.";
   }
};
class Rectangle: public Polygon
{
   public:
   Rectangle(){}
   void print(){
      cout<<"\nRectangle is a Polygon.";
   }
};
class Triangle: public Polygon
{
   public:
   Triangle(){}
   void print(){
      cout<<"\nTriangle is a Polygon.";
   }
};
```

```
class Square: public Rectangle
{
   public:
   Square(){}
   void print(){
      cout<<"\nSquare is a Rectangle.";
   }
};
int main()
{
   Shape S;
   Polygon P;
   Rectangle R;
   Triangle T;
   Square Sq;
   S.print();
   P.print();
   R.print();
   T.print();
   Sq.print();
   return 0;
}
```



Output:

This is a shape.
Polygon is a shape.
Rectangle is a Polygon.
Triangle is a Polygon.
Square is a Rectangle.

6. Suppose we have three classes Vehicle, FourWheeler, and Car. The class Vehicle is the base class, the class FourWheeler is derived from it and the class Car is derived from the class FourWheeler. Class Vehicle has a method 'vehicle' that prints 'I am a vehicle', class FourWheeler has a method 'fourWheeler' that prints 'I have four wheels', and class Car has a method 'car' that prints 'I am a car'. So, as this is a multi-level inheritance; we can have access to all the other class methods from the object of the class Car. We invoke all the methods from a Car object and print the corresponding outputs of the methods. So, if we invoke the methods in this order, car(), fourWheeler(), and vehicle(), then the output will be

I am a car
I have four wheels
I am a vehicle

Write a C++ program to demonstrate multilevel inheritance using this.

```
// mlevel.cpp
#include <iostream>
using namespace std;
class Vehicle
{
   public:
     vehicle()
     {
        cout<<"I am a vehicle\n";
     }
};
class FourWheeler : public Vehicle
{
   public:
     fourWheeler()
     {
        cout<<"I have four wheels\n";
     }
};
class Car : public FourWheeler
{
   public:
      Car()
     {
        cout<<"I am a car\n";
     }
};

int main()
{
   Car obj;
   obj.car();
   obj.fourWheeler();
   obj.vehicle();
   return 0;
}
```

Output:

```
I am a car
I have four wheels
I am a vehicle
```