WAP to print multiplication table of a given number.

```
#include<iostream>
using namespace std;
#include<conio.h>
main(){
        int n,i,table;
        cout<<"Enter the value of n:";
        cin>>n;
        for(i=1;i<=10;i++){

                table=n*i;
        cout<<n<<"*"<<i<<"="<<table<<endl;
        }
       getch();
       }
```

WAP to input p,t,r and calculate the simple interest.

```
#include<iostream>
using namespace std;
#include<conio.h>
main(){
        float p,t,r,s;
        cout<<"enter the value of p,t,r:";
        cin>>p>>t>>r;
        s=(p*t*r)/100;
       cout<<"simple interest is="<<s;
       getch();
       }
```

WAP to input two different length and breadth of two different rectangles Calculate the difference between the area and perimeter separate.

```
#include<iostream>
using namespace std;
#include<conio.h>
main(){
    float l1,l2,b1,b2,a1,a2,p1,p2,a,p;
  cout<<"enter the length and breadth of
      first rectangle:";
  cin>>l1>>b1;
  cout<<"enter the length and breadth of
      second rectangle:";
 cin>>l2>>b2;
 a1=l1*b1;
 a2=l2*b2;
 p1=2*(l1+b1);
 p2=2*(l2+b2);
 a=a1-a2;
 b=p1-p2;
 cout<<"difference of area="<<a<<endl
 cout<<"difference of perimeter="<<p;
 getch();
 }
```

→ Class and Objects

Features of C++
1. Abstraction
2. Data hiding
3. Encapsulation
4. Inheritance
5. Polymorphism

Assignment:

Wap to create a class called person that has name and age of a person. Write a necessary member function to input data and display them

```
#include<iostream>
using namespace std;
#include<conio.h>
class person{
        private:
            char name[30];
            int age;
            void input(){
            cout<<"enter the name of person:"
            <<endl;
          cin>>name;
         cout<<"enter the age of the person:"<<
          endl;
         cin>>age;
         }
public:
     void display(){
```

```
    input();
    cout<<"name of the person="<<name<<
        endl;
    cout<<"age of the person="<<age;
    }
};
main(){
    person p;
    p.display();
    getch();
    }
```

Wap to input two numbers from keyboard and display larger value create a class and membervas per your choice function.

```
#include<iostream>
#include using namespace std;
#include<conio.h>
class large{
        private:
            int a,b;
            void input(){
                cout<<"enter the value of
                    a and b:"<<endl;
                cin>>a>>b;
            }
        public:
            void display(){
                input();
                if(a>b)
```

```
        cout<<"larger value is="<<a<<endl;
    else
        cout<<"larger value is="<<b<<endl;
            }
    };

main(){
    largek;
    k.display();
    getch();
    }
```

Wap to read the values of a, b, c and display value of x where x = a/b - c.

```
#include<iostream>
using namespace std;
#include<conio.h>
class sum{
    private:
        float a,b,c,x;
    public:
        void input(){
            cout<<"enter the values of a,b,
                c:"<<endl;
            cin>>a>>b>>c;
            }
        void display(){
            x = (a/b) - c;
            cout<<"value of x is"<<x;
            }
    };
```

```
main(){
    sum s;
    s.input();
    s.display();
    getch();
    }
```

WAP ~~that~~ will ask for a temperature in degree ~~farenhite~~ Farenhite and display it degree ~~centigr~~ celsius create class and function accordingly.

```
#include<iostream>
using namespace std;
#include<conio.h>
class temp{
   private:
        float c,f;
       void input(){
        cout<<"enter degree fahren-
         heit temperature:"<<endl;
        cin>>f;
       }
   public:
       void cal(){
              input();
            c=5*(f-32)/9;
          cout<<"converted degree
            celsuis:"<<c;
          }
};
```

```
main(){
    temp t;
    t.cal();
    getch();
}
```

* Accessing Class Members;

- Usually class members are accessed with dot(.) operator with object name

Syntax:

```
object_name.memberfunction();
object_name.datamember;
```

here, it is assumed that both the member functions and data member are also in public section.

Passing Object as an Argument:- V.V.I.P.

1st Method

```
#include<iostream>
usingnamespace std;
#include<conio.h>
class complex{
        int i,j;
      public:
        void get(){
                cout<<"Enter co-efficients of
                    i and j:";
              cin>>i>>j;
            }
        void display(){
                cout<<i<<"i+"<<j<<"j "<<endl;
                }
```

```
void add(complex c1, complex c2)
{
   r = c1.r + c2.r;
   j = c1.j + c2.j;
}
};

main(){
      complex c1, c2, c3;
      c1.get();
      c2.get();
      c1.display();
      c2.display();
      c3.add(c1, c2);
      c3.display();
      getch();
      }
```

2nd method:

```
#include<iostream>
using namespace std;
#include<conio.h>
class complex{
          int r, j;
        public:
          void get(){
            cout<<"Enter the coefficient of r
                  and j:";
            cin>>r>>j;
          }
```

```
void display(){
        cout<<r<<"r+"<<j<<"j";
        }
void add(complex c2){
        r = r+c2.r;
        j = j+c2.j;
        }
main(){
    complex c1,c2;
    c1.get();
    c2.get();
    c1.display();
    c2.display();
    c1.add(c2);
    c1.display();
    getch();
    }
```

//wap to add two distances given in km and m by passing object as an argument

```
#include<iostream>
using namespace std;
#include<conio.h>
class sum{
        int km,m;
    public:
    void input(){
        cout<<"enter two distances in
            km and m:";
        cin>>km>>m;
        }
```

```
void display(){
        cout<<km<<"km"<<m<<"m";
          }

void add(sum s3){

        m = m+s3.m;
        km = km+s3.km+m/1000;
        m = m%1000;
         }
        };
main(){
        sum s1,s2;
        s1.input();
        s2.input();
        s1.display();
        s2.display();
        s1.add(s2);
         s1.display();
         getch();
          }
```

→ Passing and returning object:-

Eg:

```
class complex{
        int i,j;
        public:
            void get(){
                cout<<"Enter co-efficients of
                        i and j :;
                cin>>i>>j;
                }
            void display(){
            cout<<i<<"i+"<<j<<"j"<<endl;
             }

complex add(complex c1, complex c2){
                    complex c;
                    c.i = c1.i + c2.i;
                    c.j = c1.j + c2.j;
                    return c;
                    }
                    };
main(){
      complex c1, c2, c3, c4;
      c1.get();
       c2.get();
       c1.display();
       c2.display();
       c4 = c3.add(c1, c2);
       c4.display();
       getch();
        }
```

Alternative method:

```
class complex{
        int i,j;
        public:
            void get(){
                cout<<"Enter co-efficient
                 of ~~cin~~ and j:";
                cin>>i>>j;
              }

    Void display(){
          cout<<i<<"i+"<<j<<"j"<<endl
              }

    complex add(complex c2)
              {
                complex c;
                c.i = i+c2.i;
                c.j = j+c2.j;
                return c;
              };}
main(){
        complex c1, c2;
        c1.get();
        c2.get();
        c1.display();
        c2.display();
        c2 = c1.add(c2);
        c2.display();
        getch();
          }
```

WAP to add two distances in km and m by passing and returning object:-

```
class sum{
        int km,m;
        public:
          void input(){
          cout<<"enter the two distances
              in km and m:";
          cin>>km>>m;
          }

          void display(){
            cout<<km<<"km"<<m<<"m";
                }
                                    ,sum s4
        sum ~~complex~~ add(sum s3){
                sum s;
                s.km=s3.m+s4.m;
                s.km=s3.km+s4.km+s.m/1000;
                s.m = s3..m %.1000;
                ~~returns~~;
              }
              };

main(){
      sum s1,s2,s3,s4;
      s1.input();
      s2. input();
      s1.display();
      s2. display();
      s4=s3.sum(s1,s2);
      s4.display();
      getch();
      }
```

* Friend Function (V.V.V.V.— Imp)

(a) By making outside (non-member) function as a fr
nd function.

```
eg: class demo{
        int a, b;
    public:
        void get(){
            cout<<"Enter a and b.
            cin>>a>>b;
            }

        friend void sum(demo d1);
        };

void sum(demo d1){
        cout<<"sum="<<d1.a+d1.b;
            }

main(){
        demo d1;
      d1.get();
      sum(d1);
```

//wap to add two times given in hour and minute by passing and returning object.

//wap to add two times given in hours, minute and sec by passing and returning object.

//wap to add two heights given in feet and inches by passing and returning object.

//wap to input ~~two~~ three values and find the highest one by using concept of friend function. Use appropriate class with member function

//wap to add objects of two different classes using outside function as a friend function for both the classes.

```
#include<iostream>
using namespace std;
#include<conio.h>
class time{
        int hr, m;
        public:
          void get(){
                cout<<"enter the time in
                  hour and minute:"<<endl;
                cin>>hr>>m;
                }
        void display(){
                cout<<hr<<"hr"<<m<<"m";
                }
```

~~complex~~ time add(time t_1, time t_2){

```
        time t;
        t.m = t1.m + t2.m;
        t.hr = t1.hr + t2.hr + t.m/60;
        t.m = t.m % 60;
        return t;
    }
};

main(){
    time t1, t2, t3, t4;
    t1.get();
    t2.get();
    t1.display();
    t2.display();
    t4 = t3.add(t1, t2);
    t4.display();
    getch();
    }
```

(2)

```
#include<iostream>
using namespace std;
#include<conio.h>
class time{
    ~~void get(){~~
        int hr, m, s;
    public:
        void get(){
            cout<<"enter time in
                hour, minute, sec:";
```

```
cin >> hr >> min m >> s;
}

void display (){

        cout << hr << "hr" << m << "m" << s << "s" <<
            endl;
        }

time add (time t1, time t2){
            time t;
            t.m = t1.m + t2.m
            t.s = t1.s + t2.s
            t.m = t1.m + t2.m + t.s/60;
            t.hr = t1.hr + t2.hr + t.m/60;
            t.s = t.s %. 60;
            t.m = t.m %. 60;
            return t;
            }
            };

main (){
        time t1, t2, t3, t4;
        t1.get();
        t2.get();
        t1.display();
        t2.display();
        t4 = t3. add (t1, t2);
        t4.display();
        getch();
        }
```

A program to add two complex numbers using friend function.

```
#include<iostream>
using namespace std;
#include<conio.h>
class complex{
        int r, j;
        public:
            void get()
            {
            cout<<"Enter co-efficients of r & j:";
            cin>>r>>j;
            }
           void display(){
               cout<<r<<"r+"<<j<<"j"<<endl;
            }
        friend complex add (complex c1, complex c2);
        }
complex add(complex c1 , complex c2){
            complex c;
            c.r = c1.r + c2.r;
            c.j = c1.j + c2.j;
            return c;
          }
main()
{
  complex c1, c2, c3;
  c1.get();
  c2.get();
  c1.disp();
  c2.disp();
     c3 = add(c1, c2);
     c3.display();
}
```

WAP to add two ~~dista~~ times given in hr, minute and sec by using friend function.

```
#include<iostream>
using namespace std;
#include<conio.h>
class time{
        int hr,m,s;
        public:
        void get(){
          cout<<"Enter the time in hr,minute
           and sec:"<<endl;
         cin>>hr>>m>>s;
          }
      void display(){

        cout<<hr<<"hr"<<m<<"m"<<s<<"sec"
                              endl;
           }
      friend time add(time t1, time t2);
       };

main(){                  time add(time t1, time t2){
   time t1,t.              time t;
                          t.s=t1.s+t2.s;
                          t.m=t1.m+t2.m+t.s/60;
                         t.hr=t1.hr+t2.hr+t.m/60;
                         t.s=t.s%60;
                         t.m=t.m%60;
                        return t;
                        }
```

```
main(){
    time t1, t2, t3;
    t1.get();
    t2.get();
    t1.display();
    t2.display();
    t3 = add(t1, t2);
    t3.display();
    getch();
    }
```

by making a member function of one class friend of another class.

```
class A;
class B{
    int b, c;
    public:
        void get()
        {
        cout<<"Enter b & c:"
        cin>>b>>c;
        }
    void add(A o b1);
    };

class A{
    int a;
    public:
        void get()
        {
        cout<<"Enter a:";
```

```
        cin>>a;
        }
    friend void B::add(A ob1);
    };

void B::add(A ob1)
    {
    cout<<"sum="<<ob1.a+b+c;
    }

main(){
        A ob1;
        B ob2;
        ob1.get();
        ob2.get();
        ob2.add(ob1);
        getch();
        }
```

WAP to create two classes input 3 variables in one class and 2 variables in another class. Calculate the mean of those values by making a member function of one class friend of another class.

```
#include<iostream>
using namespace std;
#include<conio.h>
class second;
class first{
        int a, b, c;
        public:
         void get()
          {
          cout<<"Enter a,b & c:";
          cin>>a>>b>>c;
          }
       void mean(second ob1);
       };

class second{
        int d,e;
        public:
         void get(){
             cout<<"Enter d and e :";
             cin>>d>>e;
             }
       friend void first::mean(second ob1);
         };

void first::mean(second ob1){
```

```
    cout<<"mean="<<(a+b+c+ob1.d+
                    ob1.e)/5;
}

main(){
    first ob2;
    second ob1;
    ob2.get();
    ob1.get();
    ob2.mean(ob1);
    getch();
}
```

@ making a whole class friend of another class.

eg:

```
class B;
class A {
        int a;
        public:
            void get()
            {
            cout<<"Enter a:";
            cin>>a;
            }
        friend class B;
        };

class B {
        int b;
        public:
            void get()
            {
            cout<<"Enter b:";
            cin>>b;
            }
            void add(A ob1)
            {
             cout<<"sum="<<b+ob1.a;
            }
            void diff(A ob1)
            {
            cout<<"Difference="<<ob1.a-b;
            }
        };
```

```
main(){
        A ob1;
        B ob2;
      . ob1.get();
        ob2.get();
        ob2.add(ob1);
        ob2.diff(ob1);
        }
```

WAP to create two classes. Input marks of five subjects in one class and calculate the total marks and percentage in another class by using two different functions. Also decide whether the student is pass or fail. Use the concept of friend class.

Inline functions

```
#include<iostream>
using namespace std;
#include <conio.h>
class Demo {
        int a,b;
        public:
            void get(){
              cout<<"Enter value of a
              and b:"<<endl;
              cin>>a>>b;
              }
            void calc();
        };
inline void Demo::calc(){
              cout<<"sum="<<a+b;
              }
main(){
        Demo d;
        d.get();
        d.calc();
        getch();
        }
```

Inline function is a function that is expanded in line when it is called.
When the inline function is called whole code of the inline functions gets inserted or substituted at the point of inline function call.

Default Arguments:

A default argument is a value provided in a function declaration that is automatically assigned by the compiler.

```
#include<iostream>
using namespace std;
#include<conio.h>
class demo{
        public:

        void get(int a=1,int b=2, int c=3);

        };

void demo::get(int a, int b, int c){
            cout<<"sum="<<a+b+c;
            }

main(){
        demo d;
        d.get();
        getch();
        }
```

WAP to calculate simple interest by using default value of r = 8%. calculate the SI for 10 costumers. Implement the appropriate member function with class.

```
#include<iostream>
using namespace std;
#include<conio.h>
class interest {
        public: float p,t; float p,t;
                public:
                  void get(){
                  void get(r=8){float
    void interest::void(::void get(int r=8);
          };

void interest:: calc(int r=8){
          cout<<

        float p,t;
        public:
            void input();
            void get(float r=8);
              };

    void interest::get(float r){
        cout<<"SI="<<(p*t*r)/100;
              }
    void interest::input(){
        cout<<"Enter principle and time:";
        cin>>p>>t;
              }
```

```
main(){
        interest s;
       s.input();
       s.get();
       getch();
        }
```

Alternative method:

```
class SI{
        public:
          void get(float p, float t, float r=8);
           };
void SI::get(float p, float t, float r){
          cout<<"SI ="<<(p*t*r)/100;
            }

main{
      SI s;
     s.get(1000,2);
     getch();
      }
```

Array of object:

```
class SI{
        float p,t;
        public:
            void get();
            void interest(float r=8);
             };

SI::get(){
            cout<<"enter the principle,tim
             cin>>p>>t;
              }

void SI::interest(float r=8){

             cout<<"SI="<<(p*t*r)/100;
                 }

main(){
     SI s[10];
     int i;
     for(i=0;i<10;i++){

            s[i].get();
      }
     for(i=0;i<10;i++){
                  s[i].interest();
                   }
     getch();
     }
```

eg:-

```
class Demo{
        int a, b;
        public:
            void get(){
                cout<<"Enter value of a&b:";
                cin>>a>>b;
                }
        void sum(){
                cout<<"sum="a+b;
                }
            };

main(){
    Demo d[10];
    int i;
    for(i=0; i<10; i++){
                    d[i].get();
                    }
    for(i=0; i<10; i++){
                    d[i].sum();
                    }
      getch();
        }
```

Reference variable:

WAP to swap the contents of two classes by using friend function with reference variable

```
class A{
        int a;
        public:
            void get(){
                cout<<"Enter a:";
                cin>>a;
                    }
            void dis(){
                cout<<"value of a="<<a;
                    }
            friend void swap(A &ob1, B &ob2);
                    };
class B{
        int b;
        public:
        void get(){
                cout<<"enter b:";
                cin>>b;
                    }
        void disp(){
                cout<<"value of b="<<b;
                    }
        friend void swap(A &ob1, B &ob1, B &ob2);
                    };
void swap(A &ob1, B &ob2){
                int temp;
                temp=ob1.a;
```

```
    ob1.a = ob2.a;
    ob2.b = temp;
    }

main(){
    A ob1;
    B ob2;
    ob1.get();
    ob2.get();
    Cout<<"Before swapping:";
    ob1.display();
    ob2.display();
    swap(ob1,ob2);
    cout<<"After swapping:";
    ob1.disp();
    ob2.disp();
    }
```