LAB 6

**1. WAP to find multiplication of two numbers using the concept of return by object. Show with**

**a) return by value**

**b) return by refernece**

**c) return by pointer**

**a) return by value**

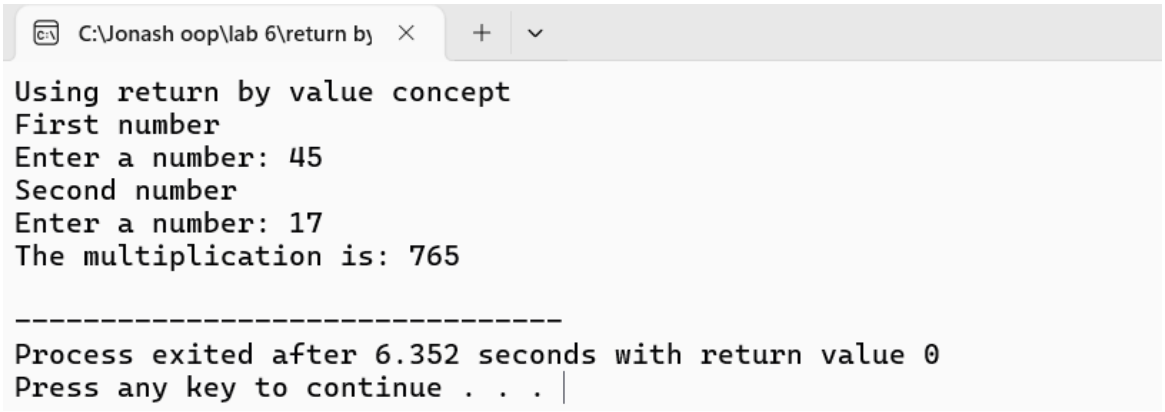
```
#include <iostream>
using namespace std;
class multiply
{
   int n;
public:
   void get()
   {
      cout << "Enter a number: ";
      cin >> n;
   }

   multiply multi(multiply x, multiply y)
   {
      multiply mul;
      mul.n = x.n * y.n;
      return mul;
   }

   void display()
   {
      cout << n << endl;
   }
};

int main()
{
   multiply a, b, c;
   cout << "Using return by value concept" << endl;
   cout << "First number" << endl;
   a.get();
   cout << "Second number" << endl;
   b.get();
   c = c.multi(a, b);
   cout << "The multiplication is: ";
   c.display();
}
```

**Output:**

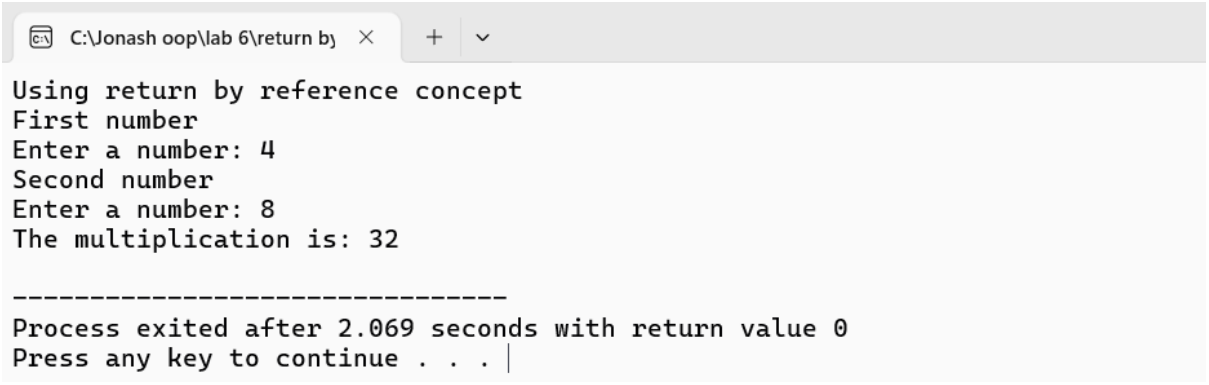



**b) return by reference**

```
#include <iostream>
using namespace std;
class multiply
{
   int n;
public:
   void get()
   {
      cout << "Enter a number: ";
      cin >> n;
   }
   multiply multi(multiply &x, multiply &y)
   {
      multiply mul;
      mul.n = x.n * y.n;
      return mul;
   }
   void display()
   {
      cout << n << endl;
   }
};

int main()
{
   multiply a, b, c;
   cout << "Using return by value concept" << endl;
   cout << "First number" << endl;
   a.get();
   cout << "Second number" << endl;
   b.get();
   c = c.multi(a, b);
   cout << "The multiplication is: ";
   c.display();
}
```

**Output:**

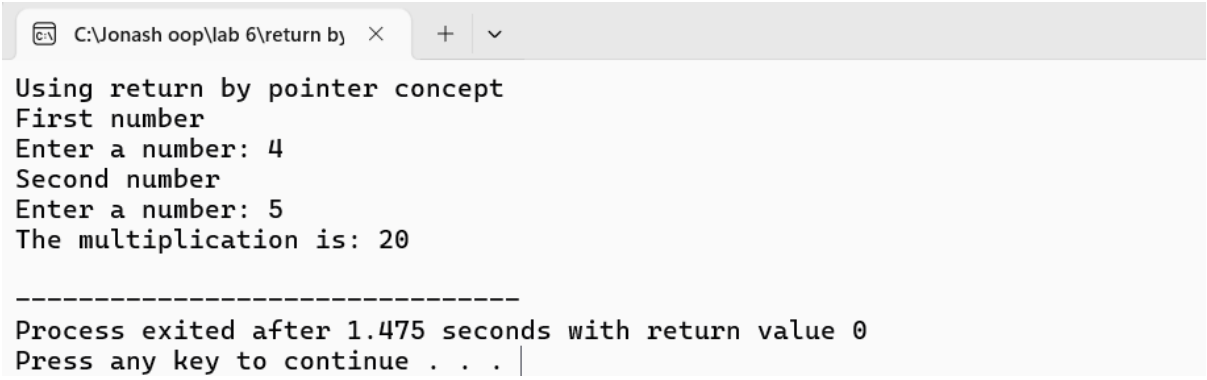



**c) return by pointer**

```
#include <iostream>
using namespace std;
class multiply
{
   int n;
public:
   void get()
   {
      cout << "Enter a number: ";
      cin >> n;
   }
   multiply *multi(multiply *x, multiply *y)
   {
      multiply *mul;
      mul->n = x->n * y->n;
      return mul;
   }
   void display()
   {
      cout << n << endl;
   }
};

int main()
{
   multiply a, b, *c;
   cout << "Using return by value concept" << endl;
   cout << "First number" << endl;
   a.get();
   cout << "Second number" << endl;
   b.get();
   c = c->multi(&a, &b);
   cout << "The multiplication is: ";
   c->display();
}
```

**Output:**


C:\Jonash oop\lab 6\return by
Using return by pointer concept
First number
Enter a number: 4
Second number
Enter a number: 5
The multiplication is: 20
--------------------------------
Process exited after 1.475 seconds with return value 0
Press any key to continue . . .

LAB 7

**1. WAP to create class named calculator that contains private data members x and y of integers type. Also define following member functions.**
**a) Add to return the result summing x and y.**
**b) Multi to return of multiply x and y.**
**c) Constructor to initialize object data members x and y.**
**d) Parameterized constructor to initialize the value of x and y given by user.**
**e) Copy constructor to initialize the value of another object.**
**f) In the above code make y as static data member and mul as static member function and perform operation.**

```
#include <iostream>
using namespace std;
class calculator
{
   int x;
   static int y;
public:
   calculator()
   {
      cout << "Enter two numbers:" << endl;
      cin >> x >> y;
   }

   // parametrized constructor
   calculator(int a, int b)
   {
      x = a;
      y = b;
   }

   // copy constructor
   calculator(calculator &cal)
   {
      x = cal.x;
      y = cal.y;
   }

   int add()
   {
      return x + y;
   }

   static int mul(int n1, int n2)
   {
      return n1 * n2;
   }

   int multi()
   {
      return mul(x, y);
```

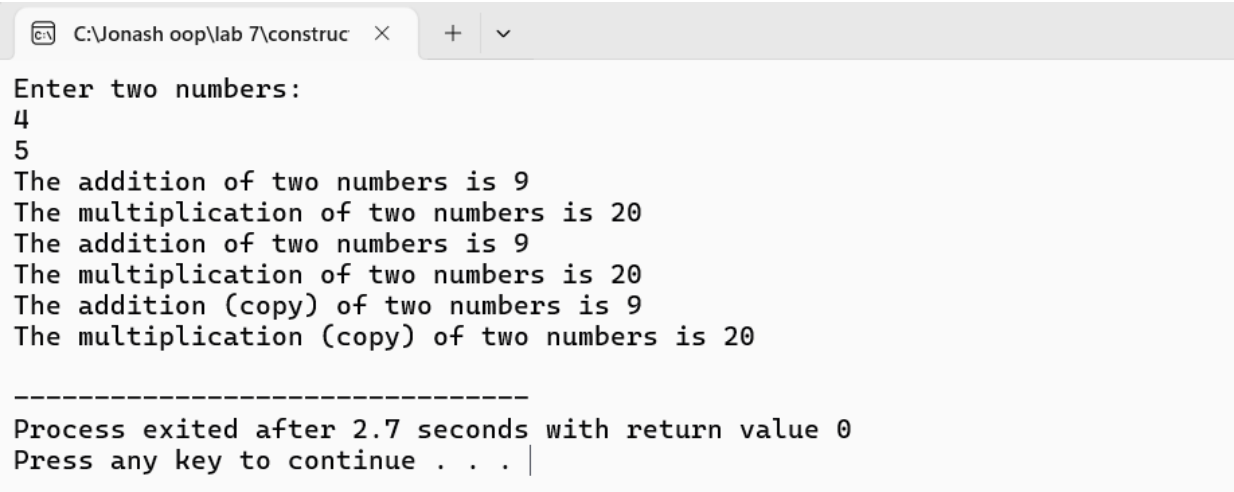
```
    }
};
int calculator::y;

int main()
{
    // default constructor
    calculator c1;
    cout << "The addition of two numbers is " << c1.add() << endl << "The multiplication of two numbers is " << c1.multi() << endl;

    // parametrized constructor
    calculator c2(4, 5);
    cout << "The addition of two numbers is " << c2.add() << endl;
    cout << "The multiplication of two numbers is " << c2.multi() << endl;

    // copy constructor
    calculator c3(c2);
    cout << "The addition (copy) of two numbers is " << c3.add() << endl;
    cout << "The multiplication (copy) of two numbers is " << c3.multi() << endl;
}
```

**Output:**

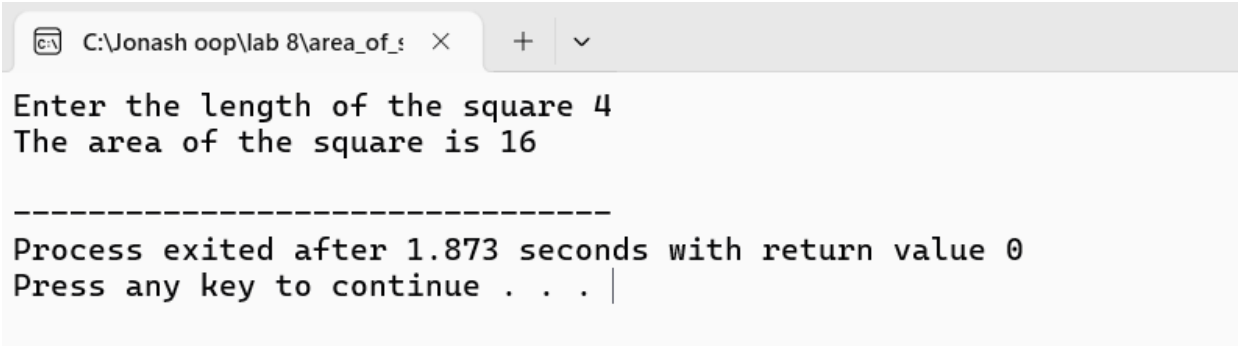


**LAB 8**

**1. WAP to find the area of square and display the result using concept of friend function.**

```
#include <iostream>
using namespace std;
class areasq
{
   int l;
public:
   areasq()
   {
      cout << "Enter the length of the square ";
      cin >> l;
   }
   int area()
   {
      return l * l;
   }
   friend void display(areasq s);
};
void display(areasq s)
{
   cout << "The area of the square is " << s.area() << endl;
}
int main()
{
   areasq a;
   a.area();
   display(a);
   return 0;
}
```

**Output:**

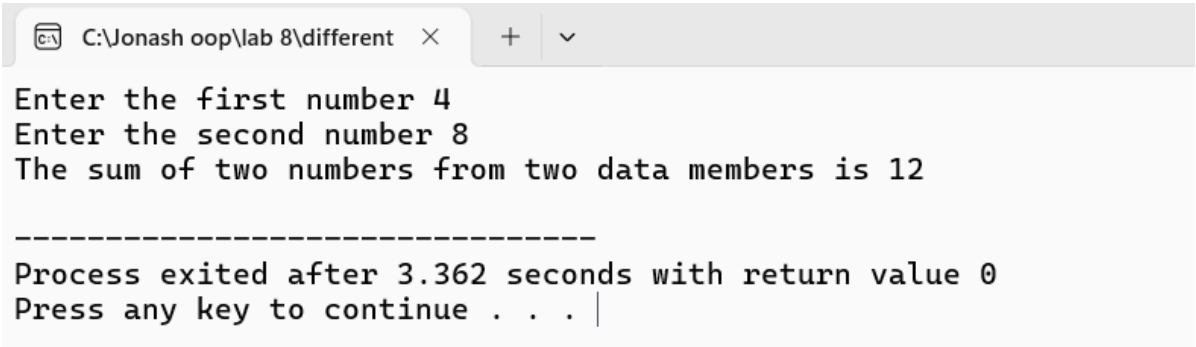



**2. WAP to find the sum of two data members of two different classes with the help of friend function.**

```
#include <iostream>
using namespace std;
class second;
// creating first class
class first
```

```
{
   int n;
public:
   first()
   {
      cout << "Enter the first number ";
      cin >> n;
   }
   friend void sum(first a, second b);
};
// creating second class
class second
{
   int n;
public:
   second()
   {
      cout << "Enter the second number ";
      cin >> n;
   }
   friend void sum(first a, second b);
};
// class friend function
void sum(first a, second b)
{
   cout << "The sum of two numbers from two data members is " << a.n + b.n << endl;
}
int main()
{
   first c1;
   second c2;
   sum(c1, c2);
   return 0;
}
```

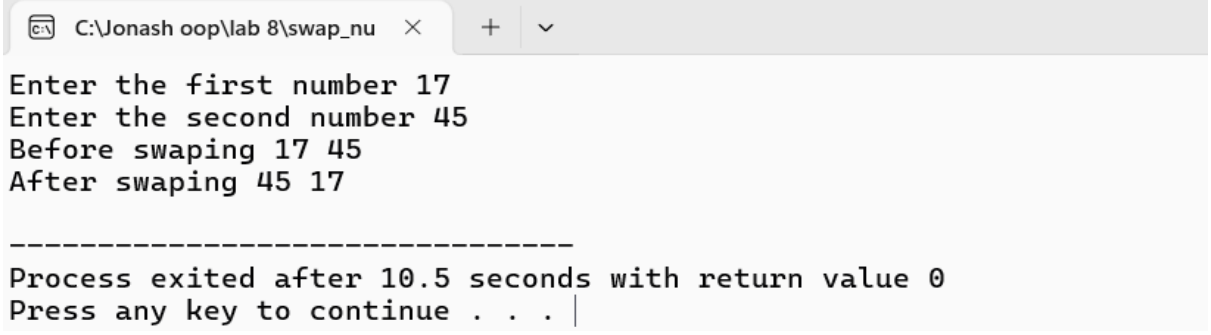
**Output:**



**3. WAP to demonstrate friend class to swap numbers of two classes.**

```
#include <iostream>
using namespace std;
class two;
class one
{
   int n;
public:
   one()
   {
      cout << "Enter the first number ";
      cin >> n;
   }
   friend void swap(one n1, two n2);
};
class two
{
   int n;
public:
   two()
   {
      cout << "Enter the second number ";
      cin >> n;
   }
   friend void swap(one n1, two n2);
};
void swap(one n1, two n2)
{
   int temp;
   cout << "Before swaping " << n1.n << " " << n2.n << endl;
   temp = n2.n;
   n2.n = n1.n;
   n1.n = temp;
   cout << "After swaping " << n1.n << " " << n2.n << endl;
}
int main()
{
   one a;
   two b;
   swap(a, b);
}
```

**Output:**



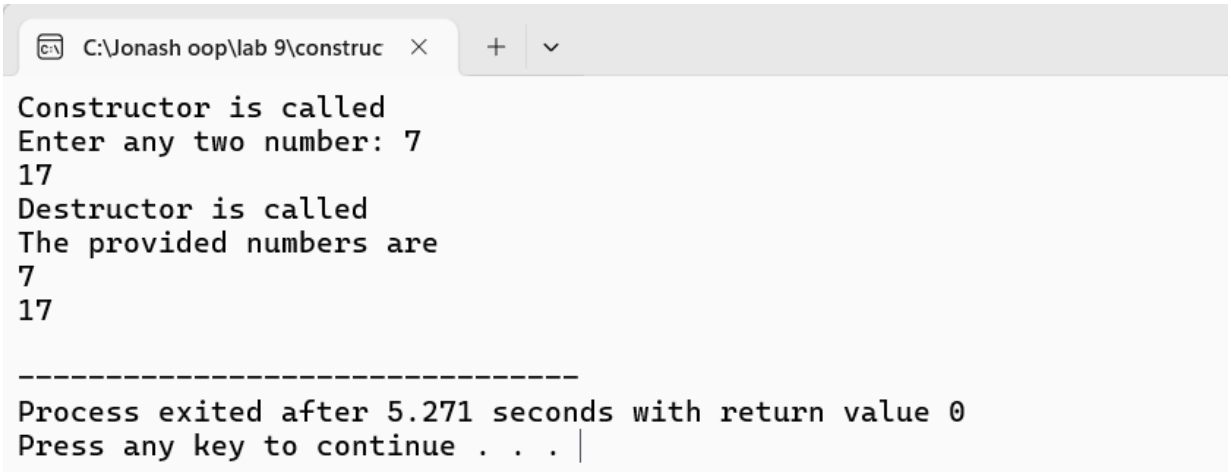
LAB 9

**1. WAP to show constructor and desctructor.**

```
#include <iostream>
using namespace std;
class oop
{
   int x, y;
public:
   oop()
   {
      cout << "Constructor is called" << endl << "Enter any two number: ";
      cin >> x >> y;
   }
   ~oop()
   {
      cout << "Destructor is called" << endl<< "The provided numbers are" << endl<< x << endl<<
y << endl;
   }
};

int main()
{
   oop a;
   return 0;
}
```

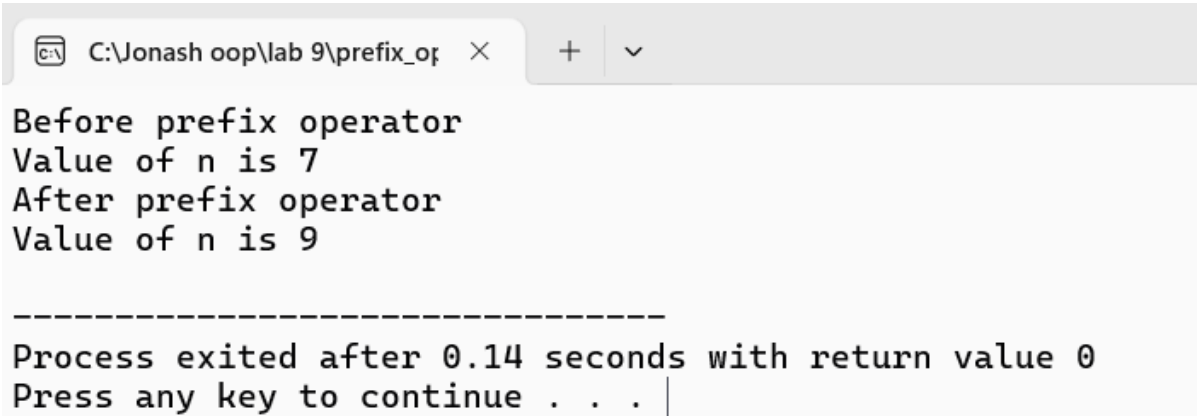
**Output:**



**2. WAP to overload ++ (prefix) operator to increase data members by 2.**

```
#include <iostream>
using namespace std;
class increment
{
   int n = 7;
public:
   increment operator++()
   {
      increment a;
      n = n + 2;
      a.n = n;
   }
   void show()
   {
      cout << n << endl;
   }
};
int main()
{
   increment c1, c2;
   cout << "Before prefix operator" << endl<< "Value of n is ";
   c1.show();
   cout << "After prefix operator" << endl<< "Value of n is ";
   c2 = ++c1;
   c2.show();
   return 0;
}
```

**Output:**



**3. WAP to overload -- (postfix) operator to descrease data members by 5.**

```
#include <iostream>
using namespace std;
class decrement
{
   int n = 17;
public:
   decrement operator--(int)
   {
      decrement a;
      n = n - 5;
      a.n = n;
   }
   void show()
   {
      cout << n << endl;
   }
};
int main()
{
   decrement c1, c2;
   cout << "Before postfix operator" << endl<< "Value of n is ";
   c1.show();
   cout << "After postfix operator" << endl<< "Value of n is ";
   c2 = c1--;
   c2.show();
   return 0;
}
```

**Output:**