Practical 1(A)

Aim: Write a C++ program to create a simple calculator.

Algorithm:

The program takes two numbers and an operation as input from the user.

The operation is stored in a character variable operation.

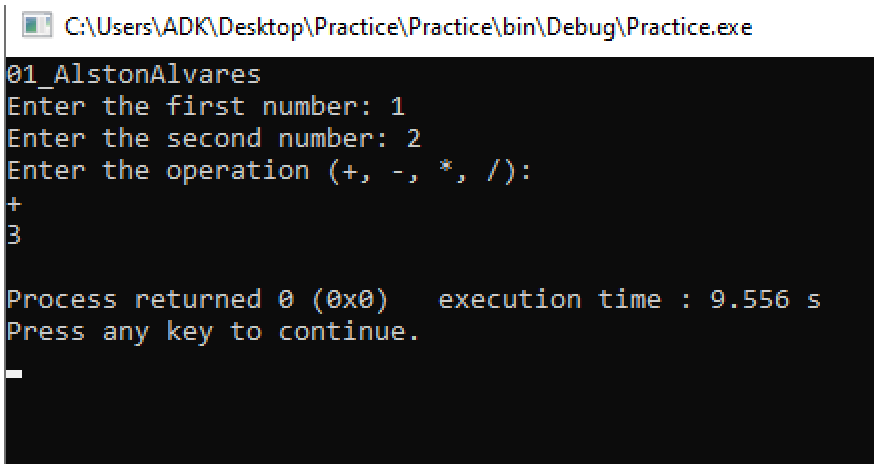
The program then uses a series of if-else statements to perform the corresponding calculation and outputs the result.

The program also includes error handling for division by zero and invalid operator.

Code:

```
#include <iostream>
using namespace std;


int main() {
   double num1, num2;
   char operation;
   cout<<"01_AlstonAlvares"<<endl;
   cout << "Enter the first number: ";
   cin >> num1;
   cout << "Enter the second number: ";
   cin >> num2;
   cout << "Enter the operation (+, -, *, /): ";
   cin >> operation;
   if (operation == '+') {
      cout << num1 + num2 << endl;
   } else if (operation == '-') {
        cout << num1 - num2 << endl;
    } else if (operation == '*') {
        cout << num1 * num2 << endl;
    } else if (operation == '/') {
        if (num2 == 0) {
            cout << "Error: division by zero" << endl;
        } else {
            cout << num1 / num2 << endl;
        }
    } else {
        cout << "Error: invalid operator" << endl;
    }
    return 0;
}
```

Output:



```
C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe
01_AlstonAlvares
Enter the first number: 1
Enter the second number: 2
Enter the operation (+, -, *, /):
+
3

Process returned 0 (0x0)   execution time : 9.556 s
Press any key to continue.
```

Conclusion: Successfully created a simple calculator.

Practical 1(B)

Aim: Write a C++ program to convert seconds into hours, minutes and seconds.

Algorithm:

The input number of seconds is first divided by 3600 to get the number of hours.

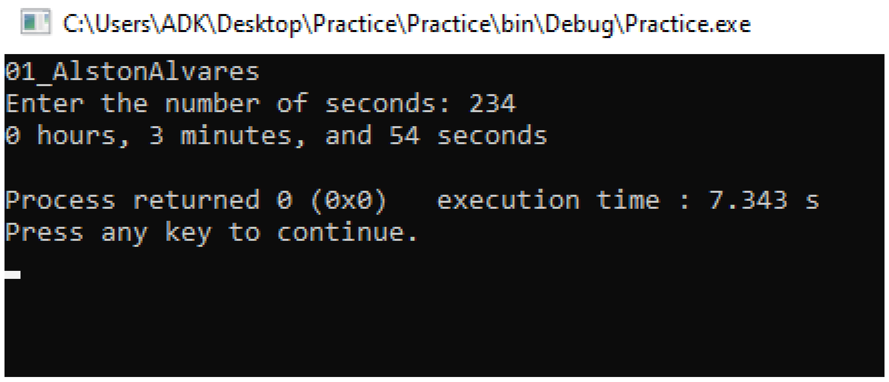
Then, the remaining seconds (after getting hours) is divided by 60 to get the number of minutes.

Finally, the remaining seconds (after getting minutes) is the number of seconds.

Code:

```
#include <iostream>
using namespace std;


int main() {
   int seconds, hours, minutes;
   cout<<"01_AlstonAlvares"<<endl;
   cout << "Enter the number of seconds: ";
   cin >> seconds;
   hours = seconds / 3600;
   minutes = (seconds % 3600) / 60;
   seconds = (seconds % 3600) % 60;
   cout << hours << " hours, " << minutes << " minutes, and " << seconds << "
seconds" << endl;
   return 0;
}
```

Output:



```
C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe
01_AlstonAlvares
Enter the number of seconds: 234
0 hours, 3 minutes, and 54 seconds

Process returned 0 (0x0)   execution time : 7.343 s
Press any key to continue.
```

Conclusion: Successfully converted seconds into hours, minutes and seconds.

Practical 1(C)

Aim: Write a C++ program to find the volume of a square, cone, and rectangle.

Algorithm:

The program starts by displaying a menu to the user, asking them to choose between finding the volume of a square, rectangle, or cone.

The user's choice is stored in the variable choice.

The program then uses a series of if-else statements to perform the corresponding calculation based on the user's choice, and outputs the result.

The program also includes error handling for invalid choice.

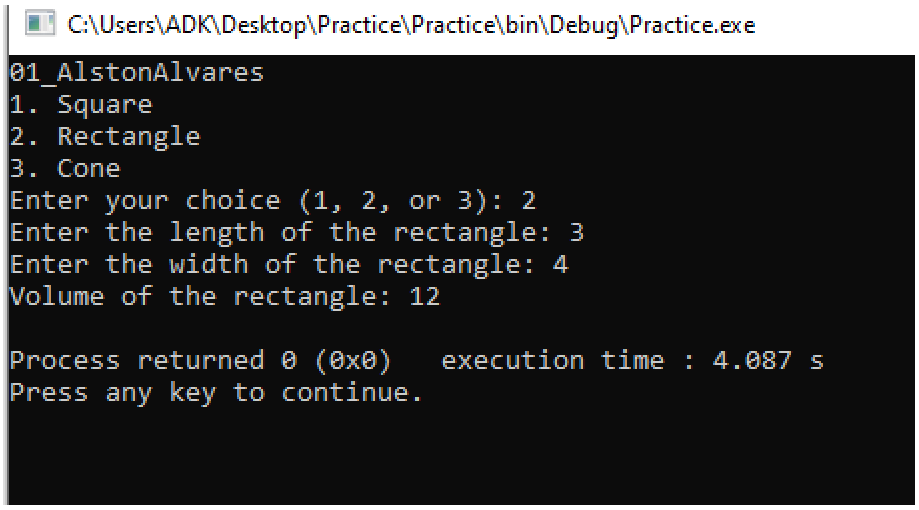
Code:

```
#include <iostream>
#include <cmath>
using namespace std;

const double pi = 3.14159265358979323846;

int main()
{
   cout<<"01_AlstonAlvares"<<endl;
   int choice;
   double a, b, h, r, l, w, volume;
   cout << "1. Square" << endl;
   cout << "2. Rectangle" << endl;
   cout << "3. Cone" << endl;
   cout << "Enter your choice (1, 2, or 3): ";
```

```
cin >> choice;
if (choice == 1)
{
   cout << "Enter the side of the square: ";
   cin >> a;
   volume = pow(a, 3);
   cout << "Volume of the square: " << volume << endl;
}
else if (choice == 2)
{
   cout << "Enter the length of the rectangle: ";
   cin >> l;
   cout << "Enter the width of the rectangle: ";
   cin >> w;
   volume = l * w;
   cout << "Volume of the rectangle: " << volume << endl;
}
else if (choice == 3)
{
   cout << "Enter the radius of the cone: ";
   cin >> r;
   cout << "Enter the height of the cone: ";
   cin >> h;
   volume = (pi * pow(r, 2) * h) / 3;
```

```
        cout << "Volume of the cone: " << volume << endl;
    }
    else
    {
        cout << "Error: invalid choice" << endl;
    }
    return 0;
}
```

Output:



```
C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe
01_AlstonAlvares
1. Square
2. Rectangle
3. Cone
Enter your choice (1, 2, or 3): 2
Enter the length of the rectangle: 3
Enter the width of the rectangle: 4
Volume of the rectangle: 12

Process returned 0 (0x0)   execution time : 4.087 s
Press any key to continue.
```

Conclusion: Successfully created a program to find the volume of a square, cone, and rectangle.

Practical 2(A)

Aim: Write a C++ program to find the greatest of three numbers.

Algorithm:

The program takes three numbers as input from the user.

The variable max is initialized with the value of the first number num1.

The program then uses a series of if statements to compare num2 and num3 with max, and updates the value of max if either num2 or num3 is greater than max.

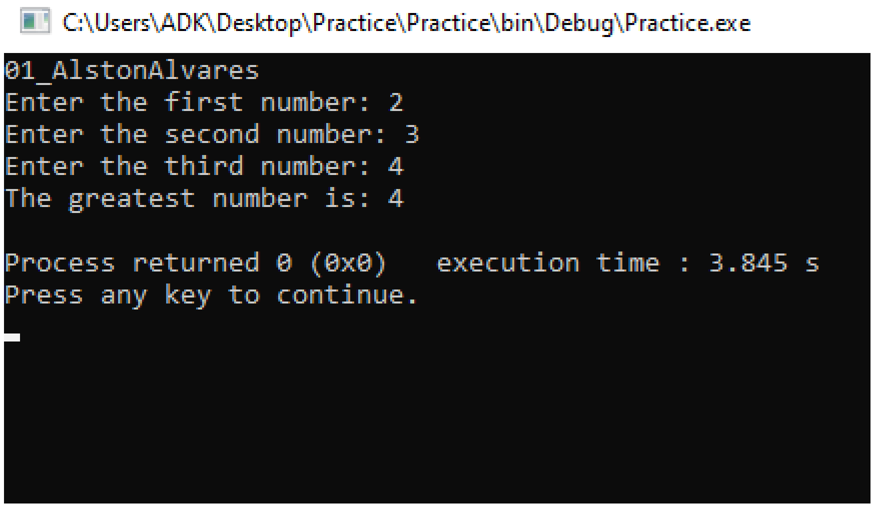
Finally, the program outputs the value of max, which is the greatest of the three numbers.

Code:

```
#include <iostream>
using namespace std;

int main() {
   cout<<"01_AlstonAlvares"<<endl;
   int num1, num2, num3;
   cout << "Enter the first number: ";
   cin >> num1;
   cout << "Enter the second number: ";
   cin >> num2;
   cout << "Enter the third number: ";
   cin >> num3;
   int max = num1;
   if (num2 > max) {
      max = num2;
```

```
    }
    if (num3 > max) {
        max = num3;
    }
    cout << "The greatest number is: " << max << endl;
    return 0;
}
```

Output:



C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe

```
01_AlstonAlvares
Enter the first number: 2
Enter the second number: 3
Enter the third number: 4
The greatest number is: 4

Process returned 0 (0x0)   execution time : 3.845 s
Press any key to continue.
```

Conclusion: Successfully created a program to find the greatest of three numbers.

Practical 2(B)

Aim: Write a C++ program to find the sum of even and odd n natural numbers.

Algorithm:

The program takes a positive integer n as input from the user.

The variables sum_even and sum_odd are used to store the sum of even and odd numbers, respectively. They are both initialized to 0.

The program uses a for loop to iterate from 1 to n, and uses an if-else statement to determine whether the current number is even or odd.

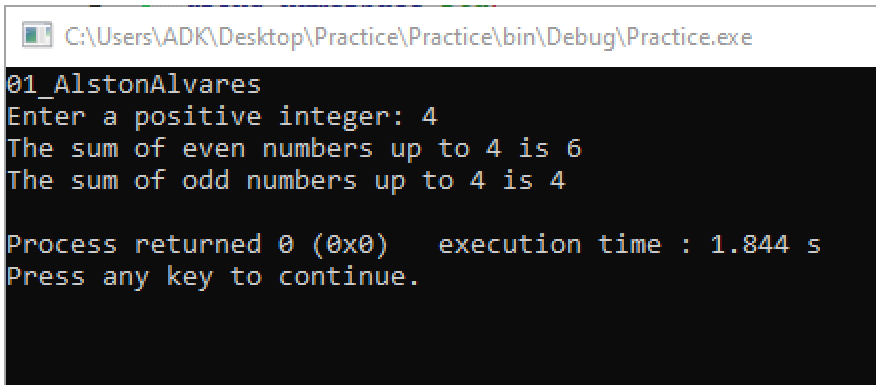
If the current number is even, it is added to sum_even. If it is odd, it is added to sum_odd.

Finally, the program outputs the values of sum_even and sum_odd, which are the sum of even and odd numbers from 1 to n, respectively.

Code:

```
#include <iostream>
using namespace std;

int main() {
   cout<<"01_AlstonAlvares"<<endl;
  int n, sum_even = 0, sum_odd = 0;

  cout << "Enter a positive integer: ";
  cin >> n;

  // calculate the sum of even and odd natural numbers15
  for(int i = 1; i <= n; i++) {
    if(i % 2 == 0) {
      sum_even += i;
    } else {
      sum_odd += i;
    }
  }

  // display the results
  cout << "The sum of even numbers up to " << n << " is " << sum_even << endl;
  cout << "The sum of odd numbers up to " << n << " is " << sum_odd << endl;

  return 0;
}
```

Output:



C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe

```
01_AlstonAlvares
Enter a positive integer: 4
The sum of even numbers up to 4 is 6
The sum of odd numbers up to 4 is 4

Process returned 0 (0x0)   execution time : 1.844 s
Press any key to continue.
```

Conclusion: Successfully created a program to find the sum of even and odd n natural numbers.

Practical 2(C)

Aim: Write a C++ program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.

Algorithm:

The program takes a positive integer n as input from the user.

The function is_prime returns true if the input number is prime, and false otherwise. It first checks if the input number is less than or equal to 1, in which case it returns false, since 1 is not a prime number.

Then, it uses a for loop to check if the input number is divisible by any number between 2 and the square root of the input number.

If the input number is divisible by any of these numbers, the function returns false, indicating that the input number is not prime.

In the main function, the program uses a for loop to iterate from 2 to n, and calls the is_prime function on each number.

If the is_prime function returns true, the number is output as a prime number.

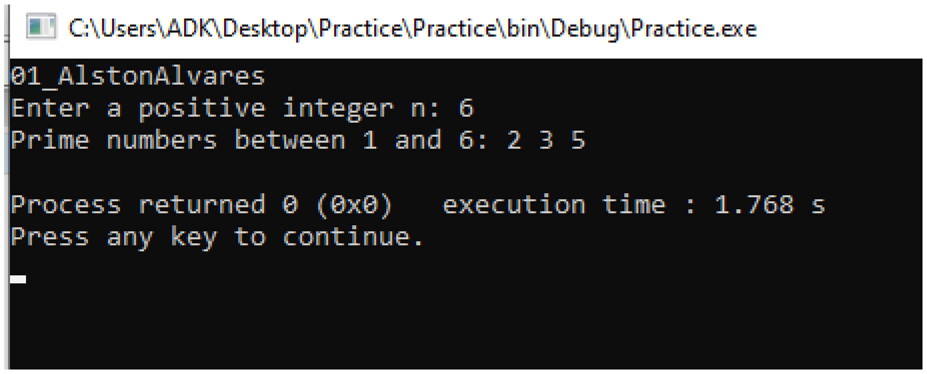
Finally, the program outputs all the prime numbers between 1 and n.

Code:

```
#include <iostream>
#include <cmath>
using namespace std;


bool is_prime(int n) {
  if (n <= 1) {
    return false;
  }
```

```
    for (int i = 2; i <= sqrt(n); i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}

int main() {
    cout<<"01_AlstonAlvares"<<endl;
    int n;
    cout << "Enter a positive integer n: ";
    cin >> n;
    cout << "Prime numbers between 1 and " << n << ": ";
    for (int i = 2; i <= n; i++) {
        if (is_prime(i)) {
            cout << i << " ";
        }
    }
    cout << endl;
    return 0;
}
```

Output:



C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe

```
01_AlstonAlvares
Enter a positive integer n: 6
Prime numbers between 1 and 6: 2 3 5

Process returned 0 (0x0)   execution time : 1.768 s
Press any key to continue.
```

Conclusion: Successfully created a program to generate all the prime numbers between 1 and n, where n is a value supplied by the user.

Practical 3(A)

Aim: Write a C++ program using classes and object Student to print name of the student, roll no. Display the same.

Algorithm:

A class is a blueprint, or prototype which defines and describes the member attributes and member functions.

Objects are the instances of a class. Simply speaking, it may be defined as a variable of user-defined datatype classes. Through an object or instance, all member variables and member functions of classes can be accessed.

For a better understanding,

Classes and Objects

Class Syntax

```
class Classname {
   access_specifier:
   member_varibales;
   member_functions;
}object1, object2;
```

Read and Print Student Information Class Example Program

Read and Print Student Information Class Example Program In C++

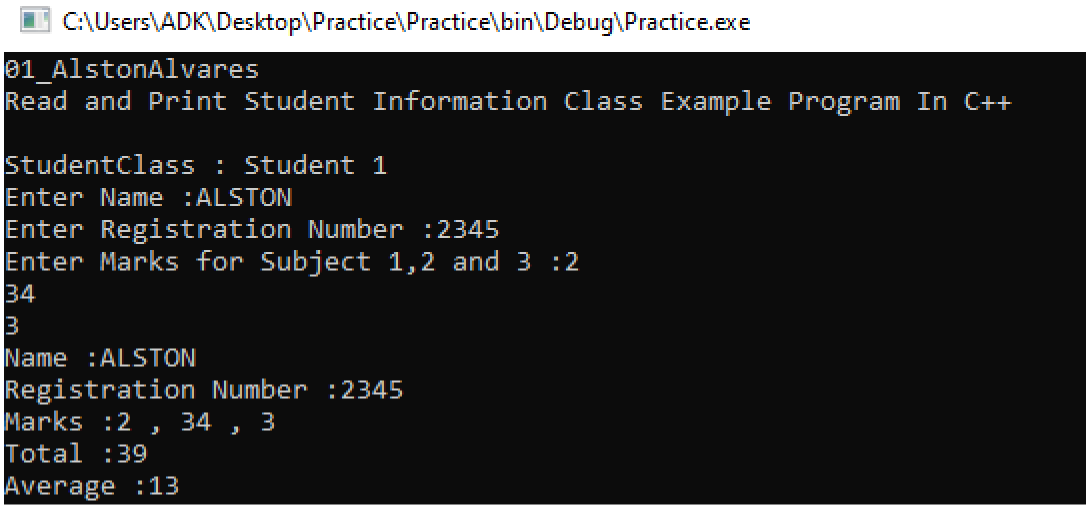
Class C++ Programs, Understanding Class,C++ Examples .

Code:

```
#include <iostream>
#include<conio.h>
```

```
using namespace std;

// Student Class Declaration
class StudentClass {
private://Access - Specifier
  //Member Variable Declaration
  char name[20];
  int regNo, sub1, sub2, sub3;
  float total, avg;

public://Access - Specifier
  //Member Functions read() and print() Declaration

  void read() {
    //Get Input Values For Object Variables
    cout << "Enter Name :";
    cin >> name;

    cout << "Enter Registration Number :";
    cin >> regNo;

    cout << "Enter Marks for Subject 1,2 and 3 :";
    cin >> sub1 >> sub2>> sub3;
  }
  void sum() {
    total = sub1 + sub2 + sub3;
    avg = total / 3;
  }

  void print() {
    //Show the Output
    cout << "Name :" << name << endl;
    cout << "Registration Number :" << regNo << endl;
    cout << "Marks :" << sub1 << " , " << sub2 << " , " << sub3 << endl;
    cout << "Total :" << total << endl;
    cout << "Average :" << avg << endl;
  }
};

int main() { cout<<"01_AlstonAlvares"<<endl;
  // Object Creation For Class
  StudentClass stu1, stu2, stu3;
  cout << "Read and Print Student Information Class Example Program In C++\n";
  cout << "\nStudentClass : Student 1" << endl;
  stu1.read();
  stu1.sum();
  stu1.print();
  cout << "\nStudentClass : Student 2" << endl;

  stu2.read();

  stu2.sum();

  stu2.print();

  cout << "\nStudentClass : Student 3" << endl;

  stu3.read();

  stu3.sum();

  stu3.print();

  getch();

  return 0;

}
```

Output:



```
C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe
01_AlstonAlvares
Read and Print Student Information Class Example Program In C++

StudentClass : Student 1
Enter Name :ALSTON
Enter Registration Number :2345
Enter Marks for Subject 1,2 and 3 :2
34
3
Name :ALSTON
Registration Number :2345
Marks :2 , 34 , 3
Total :39
Average :13
```

Conclusion: Successfully created a program using classes and object Student to print name of the student, roll_no. Display the same.

Practical 3(B)

Aim: Write a C++ program for Structure bank employee to print name of the employee, account_no. & balance. Display the same also display the balance after withdraw and deposit.

Algorithm:

In this program, we first define the BankEmployee structure, which has three members: a string for the employee's name, an int for their account number, and a float for their balance.

We then read in the employee's name, account number, and starting balance from the user, and display them using cout.

Next, we enter a loop that allows the user to make deposits and withdrawals until they choose to quit. Each time through the loop,

we display a prompt asking the user to choose 'd' to deposit, 'w' to withdraw, or 'q' to quit.

We read in their choice using cin, and then use a switch statement to handle the different cases.

If the user chooses 'd', we read in the amount to deposit, add it to the employee's balance, and display the new balance using cout.

If the user chooses 'w', we read in the amount to withdraw, check if it's greater than the employee's balance, and either display an "Insufficient funds" message or subtract the amount from the balance and display the new balance.

If the user chooses 'q', we break out of the loop and exit the program.

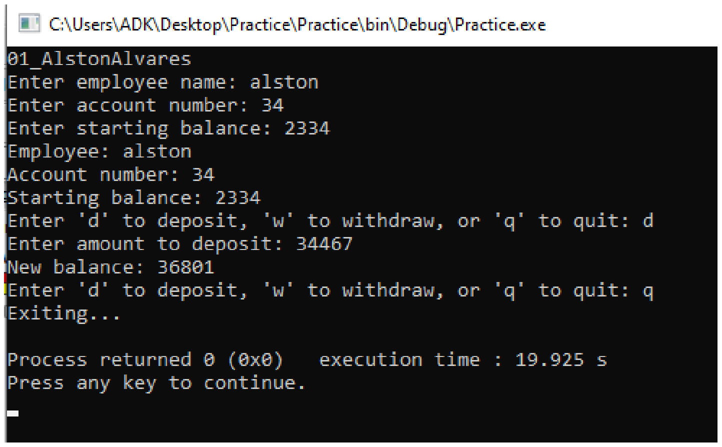
If the user enters an invalid choice, we display an error message and loop back to the prompt.

Code:

```
#include <iostream>
using namespace std;

struct BankEmployee
{
   string name;
   int account_no;
   float balance;
};

int main()
{ cout<<"01_AlstonAlvares"<<endl;
   BankEmployee employee;
   cout << "Enter employee name: ";
   getline(cin, employee.name);
   cout << "Enter account number: ";
   cin >> employee.account_no;
   cout << "Enter starting balance: ";
   cin >> employee.balance;

   cout << "Employee: " << employee.name << endl;
   cout << "Account number: " << employee.account_no << endl;
   cout << "Starting balance: " << employee.balance << endl;


   float amount;
   char choice;
   do
   {  cout << "Enter 'd' to deposit, 'w' to withdraw, or 'q' to quit: "; cin >> choice;
      switch (choice)
      {
      case 'd':
         cout << "Enter amount to deposit: ";
         cin >> amount;
         employee.balance += amount;
         cout << "New balance: " << employee.balance << endl;
         break;
      case 'w':
         cout << "Enter amount to withdraw: ";
         cin >> amount;
         if (amount > employee.balance)
         {
            cout << "Insufficient funds" << endl;
         }
         else
         {
            employee.balance -= amount;
```

```
            cout << "New balance: " << employee.balance << endl;
          }
          break;
        case 'q':
          cout << "Exiting..." << endl;
          break;
        default:
          cout << "Invalid choice" << endl;
          break;
      }
    } while (choice != 'q');

    return 0;
}
```

Output:



```
C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe
01_AlstonAlvares
Enter employee name: alston
Enter account number: 34
Enter starting balance: 2334
Employee: alston
Account number: 34
Starting balance: 2334
Enter 'd' to deposit, 'w' to withdraw, or 'q' to quit: d
Enter amount to deposit: 34467
New balance: 36801
Enter 'd' to deposit, 'w' to withdraw, or 'q' to quit: q
Exiting...

Process returned 0 (0x0)   execution time : 19.925 s
Press any key to continue.
```

Conclusion: Successfully created program for Structure bank employee to print name of the employee, account_no. & balance. Display the same also display the balance after withdraw and deposit

Practical 3(C)

Aim: Write a C++ Program to design a class having static member function named showcount() which has the property of displaying the number of objects created of the class.

Algorithm:

In this program, we first define the MyClass class, which has a private static int member named count that keeps track of the number of objects created.

We also define a constructor for the class that increments the count member each time a new object is created.

We then define a static member function named showcount(), which simply displays the current value of the count member using cout.

In the main() function, we create three objects of the MyClass class (obj1, obj2, and obj3) and call MyClass::showcount() to display the number of objects created so far.

We then create a fourth object (obj4) and call MyClass::showcount() again to display the updated count.

When this program is run, the output should be:

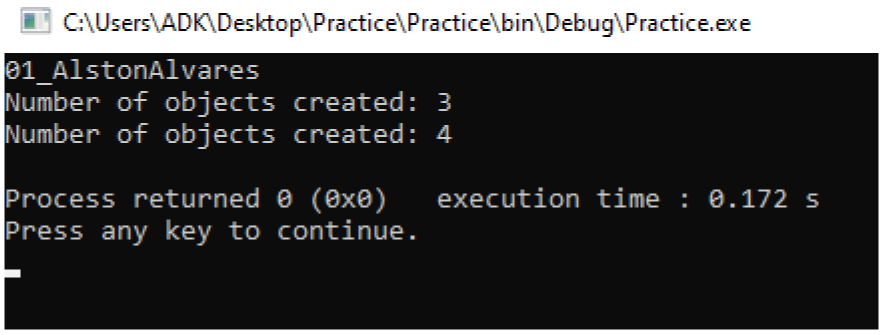
This indicates that the count member was properly incremented each time an object was created, and that the showcount() function correctly displayed the current count.

Code:

```
#include <iostream>
using namespace std;

class MyClass {
   private:
      static int count;
   public:
      MyClass() {
         count++;
      }
      static void showcount() {
         cout << "Number of objects created: " << count << endl;
      }
};

int MyClass::count = 0;

int main() { cout<<"01_AlstonAlvares"<<endl;
   MyClass obj1, obj2, obj3;

   MyClass::showcount();

   MyClass obj4;
```

```
    MyClass::showcount();


    return 0;
}
```

Output:



```
C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe
01_AlstonAlvares
Number of objects created: 3
Number of objects created: 4

Process returned 0 (0x0)   execution time : 0.172 s
Press any key to continue.
```

Conclusion: Successfully created a program to design a class having static member function named showcount() which has the property of displaying the number of objects created of the class.

Practical 3(D)

Aim: Write a Program to find Maximum out of Two Numbers using friend function. Note: Here one number is a member of one class and the other number is member of some other class.

Algorithm:

In this program, we first define ClassA, which has a private member numA, and a constructor that initializes numA with the value passed as an argument.

We also define ClassB, which has a private member numB, and a constructor that initializes numB with the value passed as an argument.

In both classes, we declare the findMax() function as a friend function, so that it can access the private members of both classes.

The findMax() function takes two arguments, objA of type ClassA and objB of type ClassB. It compares the values of numA and numB, and returns the maximum value.

In the main() function, we create an object objA of ClassA with a value of 10, and an object objB of ClassB with a value of 20.

We then call the findMax() function with objA and objB as arguments, and store the result in maxNum.

Finally, we display the maximum number using cout.

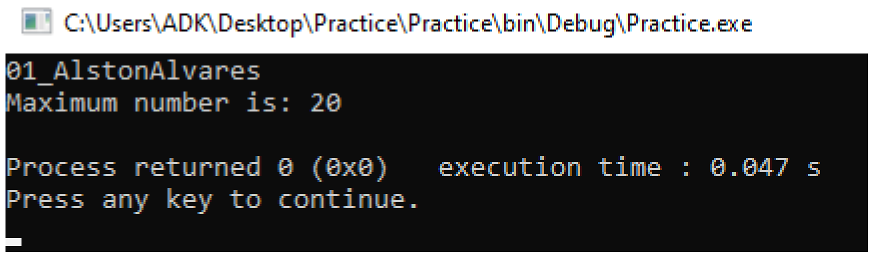
When this program is run, the output should be:

This indicates that the findMax() function correctly identified 20 as the maximum value between the two numbers.

Code:

```
#include <iostream>
using namespace std;
class ClassA; // Forward declaration of ClassA
class ClassB {
   private:
      int numB;
   public:
      ClassB(int num) {
         numB = num;
      }
      friend int findMax(ClassA objA, ClassB objB);
};
class ClassA {
   private:
      int numA;
   public:
      ClassA(int num) {
         numA = num;
      }  friend int findMax(ClassA objA, ClassB objB);
};
int findMax(ClassA objA, ClassB objB) {
   if (objA.numA > objB.numB) {
      return objA.numA;
   } else {
      return objB.numB;  } }
int main() { cout<<"01_AlstonAlvares"<<endl;
   ClassA objA(10);
   ClassB objB(20);
   int maxNum = findMax(objA, objB);
   cout << "Maximum number is: " << maxNum << endl;
   return 0;
}
```

Output:



```
C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe
01_AlstonAlvares
Maximum number is: 20

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

Conclusion: Successfully created a program to find Maximum out of Two Numbers using friend function. Note: Here one number is a member of one class and the other number is member of some other class.

Practical 3(E)

Aim: Write a C++ Program using copy constructor to copy data of an object to another object.

Algorithm:

In this program, we first define the MyClass class, which has a private member num, and a constructor that initializes num with the value passed as an argument.

We then define a copy constructor for the class, which takes a reference to another MyClass object as its argument. Inside the copy constructor, we copy the num value from the passed object to the new object, and display a message indicating that the copy constructor has been invoked.

In the main() function, we create an object obj1 of MyClass with a value of 10.

We then create a second object obj2 of MyClass, and initialize it using obj1 (i.e., we copy the data from obj1 to obj2 using the copy constructor).

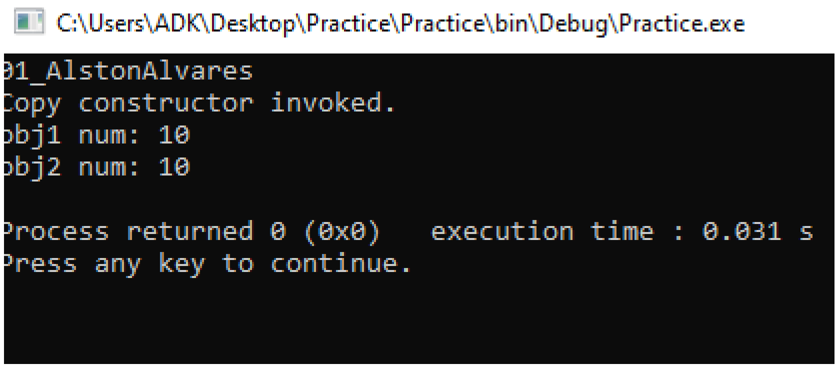
Finally, we display the values of num for both objects using the getNum() member function.

Code:

```
#include <iostream>
using namespace std;

class MyClass {
   private:
      int num;
```

```
  public:
    MyClass(int n) {
      num = n;
    }
    MyClass(const MyClass& obj) {
      num = obj.num;
      cout << "Copy constructor invoked." << endl;
    }
    int getNum() {
      return num;
    }
};

int main() { cout<<"01_AlstonAlvares"<<endl;
  MyClass obj1(10);
  MyClass obj2 = obj1;

  cout << "obj1 num: " << obj1.getNum() << endl;
  cout << "obj2 num: " << obj2.getNum() << endl;

  return 0;
}
```

Output:



```
C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe
01_AlstonAlvares
Copy constructor invoked.
obj1 num: 10
obj2 num: 10

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

Conclusion: Successfully created a program using copy constructor to copy data of an object to another object.

Practical 3(F)

Aim: Write a C++ Program to allocate memory dynamically for an object of a given class using class's constructor.

Algorithm:

In this program, we first define the MyClass class, which has a private member num, and a constructor that initializes num with the value passed as an argument.

In the main() function, we use the new operator to dynamically allocate memory for an object of MyClass with a value of 10. The new operator returns a pointer to the newly created object, which we store in the ptrObj variable of type MyClass*.

We then call the displayNum() member function on the object pointed to by ptrObj.

Finally, we use the delete operator to deallocate the memory that was previously allocated for the object.

When this program is run, the output should be:

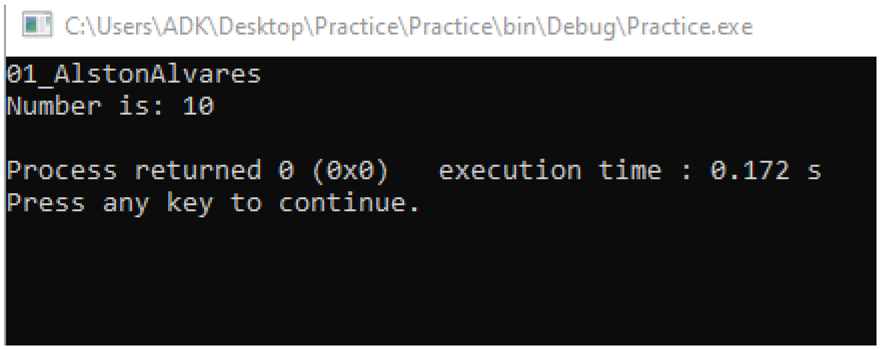
Number is: 10

This indicates that the displayNum() member function correctly displayed the value of num for the dynamically allocated object.

Code:

```
#include <iostream>
using namespace std;


class MyClass {
  private:
     int num;
  public:
     MyClass(int n) {
        num = n;
     }
     void displayNum() {
        cout << "Number is: " << num << endl;
     }
};

int main() { cout<<"01_AlstonAlvares"<<endl;
  MyClass* ptrObj = new MyClass(10);

  ptrObj->displayNum();

  delete ptrObj;

  return 0;
}
```

Output:



C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe

```
01_AlstonAlvares
Number is: 10

Process returned 0 (0x0)   execution time : 0.172 s
Press any key to continue.
```

Conclusion: Successfully created a program to allocate memory dynamically for an object of a given class using class's constructor.

Practical 4(A)

Aim: Write a C++ program to design a class representing complex numbers and having the functionality of performing addition & multiplication of two complex numbers using operator overloading.

Algorithm:

In this program, we first define the Complex class, which has two private members real and imag, and two constructors - a default constructor and a constructor that takes two arguments to initialize real and imag.

We then overload the addition operator (+) and the multiplication operator (*) as member functions of the Complex class, using the const Complex& parameter to take a reference to the second operand. Inside each overloaded operator function, we create a new Complex object to hold the result, perform the corresponding operation on the two complex numbers, and return the result.

We also define a member function display() to display the complex number in the format (real + imag*i).

In the main() function, we create three Complex objects c1, c2, and c3. We then use the overloaded addition and multiplication operators to perform the corresponding operations on c1 and c2, and store the results in c3. Finally, we display the results using the display() member function.

When this program is run, the output should be:

```
Sum: (4 + 9i)

Product: (-11 + 23i)
```

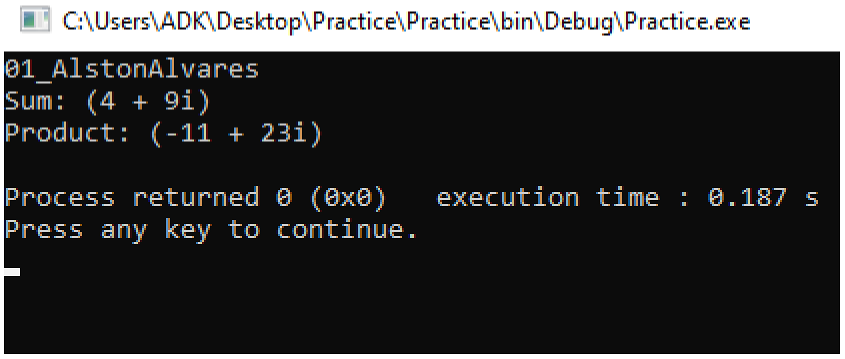
This indicates that the overloaded addition and multiplication operators correctly performed the corresponding operations on the two complex numbers.

Code:

```
#include <iostream>
using namespace std;

class Complex
{
private:
   double real;
   double imag;

public:
   Complex()
   {
      real = 0;
      imag = 0;
   }
   Complex(double r, double i)
   {
      real = r;
      imag = i;
   }
   Complex operator+(const Complex &c)
   {
      Complex result;
```

```
        result.real = real + c.real;
        result.imag = imag + c.imag;
        return result;
    }
    Complex operator*(const Complex &c)
    {
        Complex result;
        result.real = real * c.real - imag * c.imag;
        result.imag = real * c.imag + imag * c.real;
        return result;
    }
    void display()
    {
        cout << "(" << real << " + " << imag << "i)" << endl;
    }
};

int main()
{ cout<<"01_AlstonAlvares"<<endl;
    Complex c1(3, 2), c2(1, 7), c3;

    c3 = c1 + c2;
    cout << "Sum: ";
    c3.display();
    c3 = c1 * c2;

    cout << "Product: ";

    c3.display();

    return 0;

}
```

Output:



```
C:\Users\ADK\Desktop\Practice\Practice\bin\Debug\Practice.exe
01_AlstonAlvares
Sum: (4 + 9i)
Product: (-11 + 23i)

Process returned 0 (0x0)   execution time : 0.187 s
Press any key to continue.
```

Conclusion: Successfully created a program to design a class representing complex numbers and having the functionality of performing addition & multiplication of two complex numbers using operator overloading.