**SIMPLE PROGRAMS USING C++**

1. Decision Making Branching and Looping

**DATE:**

**EX.NO:1a**

**POSITIVE OR NEGATIVE**

**AIM:**

To write a C++ program for finding whether the number is positive or negative.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the variable.

**STEP 3:** Get the input from the user.

**STEP 4:** Check if (num>=0),if it is true, print the statement.

**STEP 5:** Else, print another statement.

**STEP 6:** Print the result.

**STEP 7:** stop the program.

**SOURCE CODE:**

#include<iostream>

using namespace std;

int main ()

{

int num;

cout << "Enter the number to be checked : ";

cin >> num;

if (num >= 0)

cout << num << " is a positive number.";

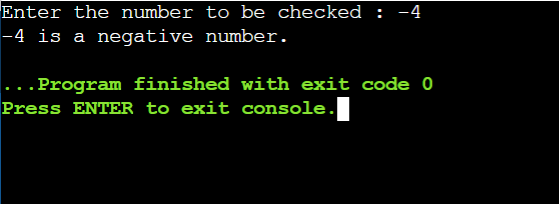
else

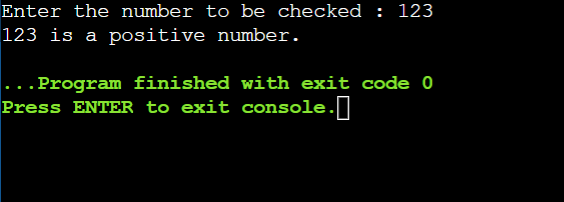
cout << num << " is a negative number.";

return 0;

}

**OUTPUT:**





**RESULT:**

Thus the C++ program for finding the number whether the number is positive or negative was executed successfully.

**DATE:**

**EX.NO:1b**

**ARMSTRONG NUMBER USING WHILE LOOP**

**AIM:**

To write a C++ program for finding Armstrong number using while loop.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the variable and initialize it.

**STEP 3:** Declare the condition inside the while loop.

**STEP 4:** Declare the condition inside the if statement, if (d==b) .

**STEP 5:** Print the result.

**STEP 6:** stop the program.

**SOURCE CODE:**

#include <iostream>

using namespace std;

int main()

{

int a,b=0,c,d;

cout<<"Enter the Number:";

cin>>a;

d=a;

while(a>0)

{

c=a%10;

b=b+(c\*c\*c);

a=a/10;

}

if(d==b)

cout<<"Armstrong Number";

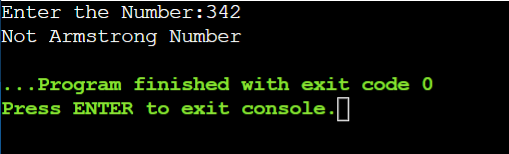
else

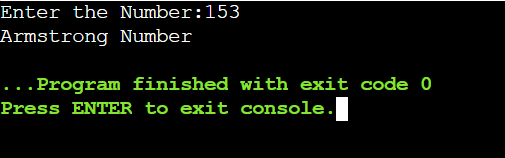
cout<<"Not Armstrong Number";

return 0;

}

**OUTPUT:**





**RESULT:**

Thus the C++ program for finding Armstrong number using while loop was executed successfully.

**DATE:**

**EX.NO:1c**

**SUM OF DIGITS**

**AIM:**

To write a C++ program for finding the sum of digits.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the variable and initialize it.

**STEP 3:** Get the input from the user.

**STEP 4:** Declare the condition inside the while loop.

**STEP 5:** Then, print the statement which is to be executed.

**STEP 6:** Print the result.

**STEP 7:** stop the program.

**SOURCE CODE:**

#include <iostream>

using namespace std;

int main()

{

int a,b=0,c;

cout<<"Enter the Number:";

cin>>a;

while(a>0)

{

c=a%10;

b=b+c;

a=a/10;

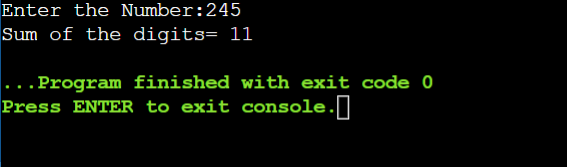
}

cout<<"Sum of the digits= "<<b;

return 0;

}

**OUTPUT:**



**RESULT:**

Thus the C++ program for finding the sum of digits was executed successfully.

**DATE:**

**EX.NO:1d**

**PALINDROME**

**AIM:**

To write a C++ program for finding whether the number is palindrome or not.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the variable and initialize it.

**STEP 3:** Get the input from the user.

**STEP 4:** Declare the condition inside the while loop.

**STEP 5:** Check if(m==b).

**STEP 6:** Print the result.

**STEP 7:** stop the program.

**SOURCE CODE:**

#include <iostream>

using namespace std;

int main()

{

int a,b=0,c,m;

cout<<"Enter the Number:";

cin>>a;

m=a;

while(a>0)

{

c=a%10;

b=(b\*10)+c;

a=a/10;

}

if(m==b){

cout<<"\n Palindrome Number";}

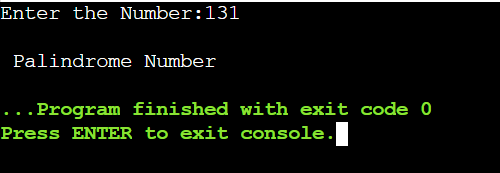
else{

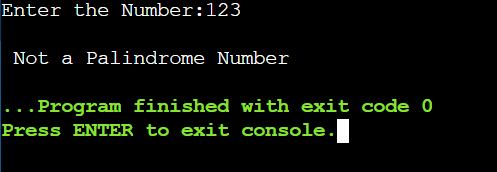
cout<<"\n Not a Palindrome Number";}

return 0;

}

**OUTPUT:**





**RESULT:**

Thus the C++ program for finding whether the number is palindrome or not was executed successfully.

**DATE:**

**EX.NO:1e**

**ARITHMETIC OPERATIONS USING SWITCH**

**AIM:**

To write a C++ program for executing Arithmetic operations using switch.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the variable and initialize it.

**STEP 3:** Declare the condition inside the switch case.

**STEP 4:** Declare the condition for addition inside the case 1.

**STEP 5:** Declare the condition for subtraction inside the case 2.

**STEP 6:** Declare the condition for multiplication inside the case 3.

**STEP 7:** Declare the condition for Division inside the case 4.

**STEP 8:** Declare the condition for Modulus inside the case 5.

**STEP 9:** Then, Print the result.

**STEP 10:** stop the program.

**SOURCE CODE:**

#include<iostream>

using namespace std;

int main()

{

float a,b,rem;

int c,d;

cout<<"Arithmetic Operatios";

cout<<"\n1.Addition\n2.Subtraction\n3.Multiplication\n4.Division\n5.Modulus";

cout<<"\n\nEnter Your Choice:\n";

cin>>c;

switch(c)

{

case 1:

{

cout<<"\nEnter the Value Of A and B ";

cin>>a>>b;

rem=a+b;

cout<<"\n Result = "<<rem;

cout<<"\n";

}

break;

case 2:

{

cout<<"\nEnter the Value Of A and B";

cin>>a>>b;

rem=a-b;

cout<<"\n Result = "<<rem;

cout<<"\n";

}

break;

case 3:

{

cout<<"\nEnter the Value Of A and B";

cin>>a>>b;

rem=a\*b;

cout<<"\n Result = "<<rem;

cout<<"\n";

}

break;

case 4:

{

cout<<"\nEnter the Value Of A and B";

cin>>a>>b;

if(a>=b)

{

rem=a/b;

cout<<"\n Result = "<<rem;

cout<<"\n";

}

else

cout<<"\nt 1st Varable Should Be Greater Than 2nd variable";

cout<<"\n";

}

break;

case 5:

{

cout<<"\nEnter the Value Of A And B";

cin>>a>>b;

if(a>=b)

{

d=a/b;

rem=a-(b\*d);

cout<<"\n Result = "<<rem;

cout<<"\n";

}

else

cout<<"\nt 1st Varable Should Be Greater Than 2nd variable";

cout<<"\n";

}

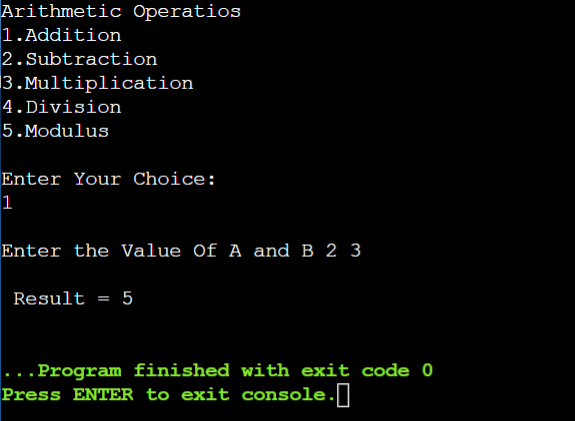
break;

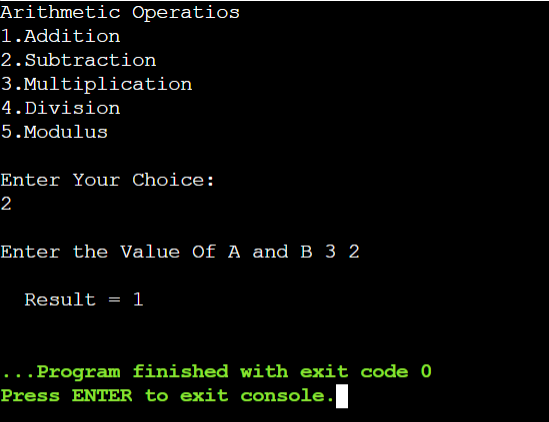
}

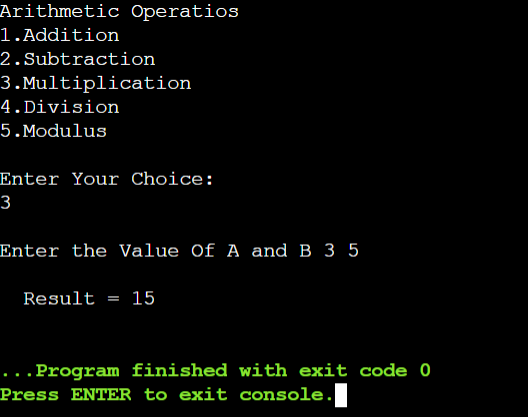
return 0;

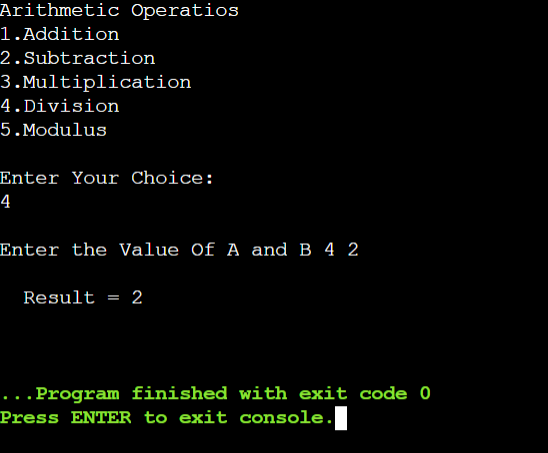
}

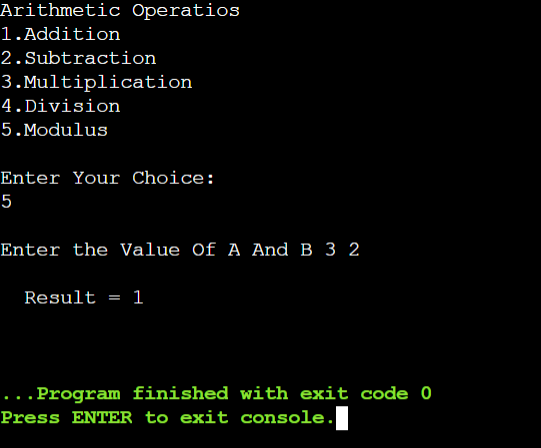
**OUTPUT:**











**RESULT:**

Thus the C++ program for executing Arithmetic operations using switch was executed successfully.

**DATE:**

**EX.NO:1f**

**LEAP YEAR OR NOT**

**AIM:**

To write a C++ program for finding whether the given year is leap year or not.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the variable.

**STEP 3:** Get the value from the user.

**STEP 4:** Declare the Nested if condition.

**STEP 5:** Then, print the statement which is to be executed.

**STEP 6:** Print the result.

**STEP 7:** stop the program.

**SOURCE CODE:**

**#**include <iostream>

using namespace std;

int main() {

int year;

cout << "Enter a year: ";

cin >> year;

if (year % 4 == 0) {

if (year % 100 == 0) {

if (year % 400 == 0)

cout << year << " is a leap year.";

else

cout << year << " is not a leap year.";

}

else

cout << year << " is a leap year.";

}

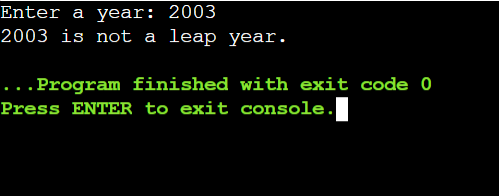
else

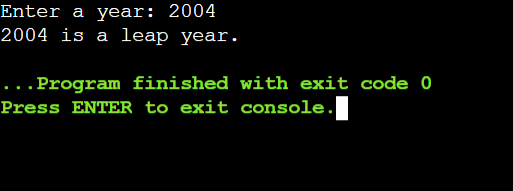
cout << year << " is not a leap year.";

return 0;

}

**OUTPUT:**





**RESULT:**

Thus the C++ program for finding whether the given year is leap year or not was executed successfully.

**DATE:**

**EX.NO:1g**

**CALCULATING ELECTRICITY BILL**

**AIM:**

To write a C++ program for Calculating Electricity bill with else if ladder.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the variable and initialize it.

**STEP 3:** Get the input for unit.

**STEP 4:** If the unit is less than or equal to 100 multiply unit with 4 and store the result in

amount.

**STEP 5:** Else if , the unit is greater than 100 and less than or equal to 300, subtract 100

from unit and multiply the remaining unit with 4.50 and add 400 to it, store the

result in amount.

**STEP 6:** Else if, the unit is greater than 300 and less than or equal to 500, subtract 300 from

unit and multiply the remaining unit with 4.75 and add 1300 to it, store the result in

amount.

**STEP 7:** Else if, the unit is greater than 500, subtract 500 from unit and multiply the

remaining unit with 5 and add 2250 to it, store the result in amount.

**STEP 8:** Print the result.

**STEP 9:** stop the program.

**SOURCE CODE:**

#include<iostream>

using namespace std;

int main()

{

int unit;

float amount;

cout<<"Enter total no.of units consumed: ";

cin>>unit;

if(unit<=100)

{

amount = 4 \* unit;

}

else if(unit>100&&unit<=300)

{

amount = 400 + ((unit - 100) \* 4.50);

}

else if(unit>300&&unit<=500)

{

amount = 400 + 900 + ((unit - 300) \* 4.75);

}

else if(unit>500)

{

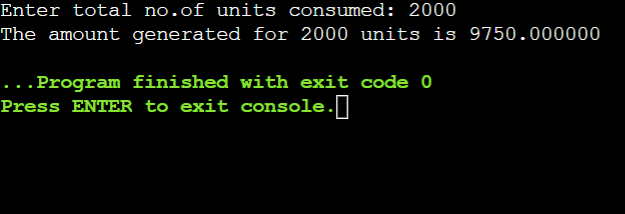
amount = 400 + 900 + 950 +((unit - 500)\*5);

}

printf("The amount generated for %d units is %f",unit,amount);

}

**OUTPUT:**



**RESULT:**

Thus the C++ program for Calculating Electricity bill with else if ladder was executed successfully.

1. **CLASSES AND OBJECTS**

**DATE:**

**EX.NO:2a**

**SIMPLE CALCULATOR**

**AIM:**

To  write a C++ program for creating  a  simple  calculator.

**ALGORITHM:**

**STEP 1:** start  the  program.

**STEP 2:** Declare  the  variable  inside  the  private  class.

**STEP 3:** Declare  the  void input() inside  the  public  class .

**STEP 4:** create  an  object  (cal).

**STEP 5 :** Create  the  menu  for  addition , subtraction ,multiplication and

                  division  using  switch  case.

**STEP 6**:  Switch  case  statement  checks  the  condition  for  the  choice

                  entered.

**STEP 7:** Print  the  result.

**STEP 8**:  stop  the  program.

**SOURCE CODE:**

#include <iostream>

using namespace std;

class calculator

{

private:

 float a,b;

public:

 void input()

 {

 cout<<"\n Enter  the  value  of  first  operand:\t";

 cin>>a;

 cout<<"\n";

 cout<<"\n Enter  the  value  of  second  operand:\t";

 cin>>b;

 cout<<"\n";

 }

 float operation1()

 {

 return a+b;

 }

 float operation2()

 {

 return a-b;

 }

 float operation3()

 {

 return a\*b;

 }

 float operation4()

 {

 return a/b;

 }

};

int main()

{

 int c;

 calculator cal;

 cout<<"\n 1.addition";

 cout<<"\n 2.subtraction";

 cout<<"\n 3.multiplication";

 cout<<"\n 4.division";

 cout<<"\n";

 cout<<"\nENTER YOUR CHOICE:\t";

 cin>>c;

 switch(c)

 {

 case 1:

 cal.input();

 cout<<"\n sum :\t"<< cal.operation1();

 break;

 case 2:

 cal.input();

 cout<<"\n Difference :\t"<<cal.operation2();

 break;

 case 3:

 cal.input();

 cout<<"\n Product :\t"<<cal.operation3();

 break;

 case 4:

 cal.input();

 cout<<"\n Quotient :\t"<<cal.operation4();

 break;

 default:

 cout<<"WRONG INPUT!!!!";

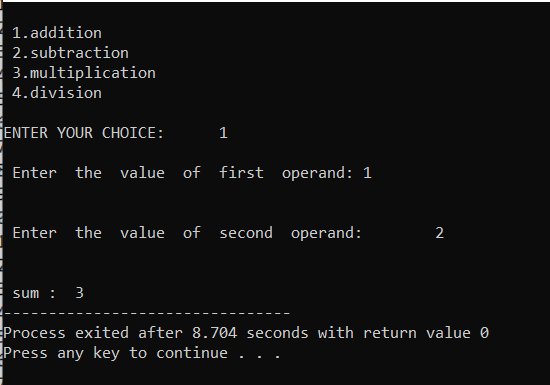
 break;

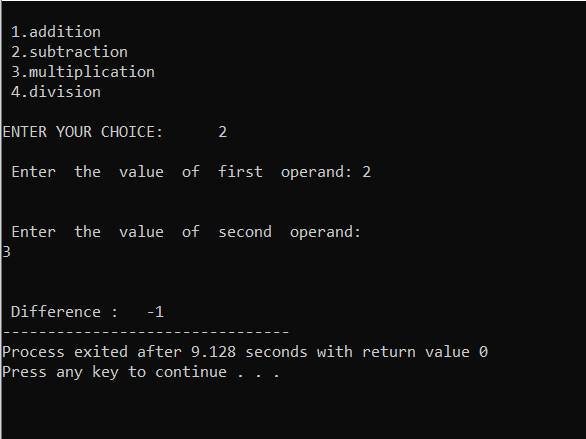
 }

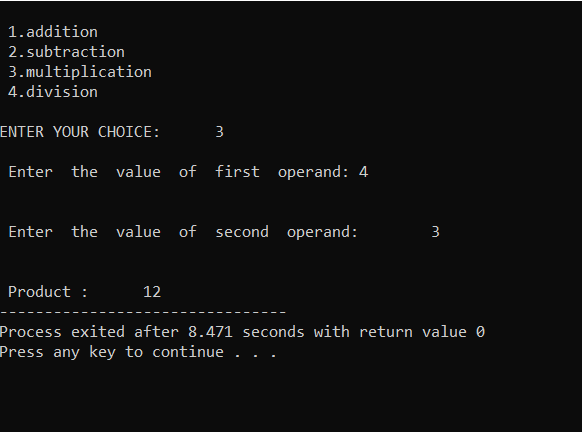
 return 0;

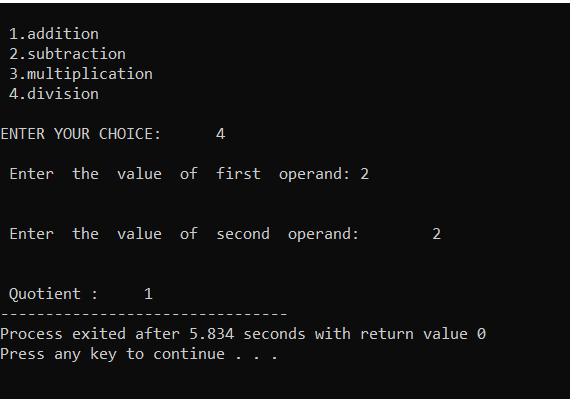
}

**OUTPUT:**









**RESULT:**

Thus the C++ program for a  simple  calculator was executed successfully.

**DATE:**

**EX.NO:2b**

**STUDENT DETAILS USING ARRAY OF OBJECTS**

**AIM:**

To write a C++ program for Creating a student details using array of objects.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the class named student.

**STEP 3:** Declare the function called void show().

**STEP 4:** The function void show() contains the data which is to be displayed.

**STEP 5:** In the main function ,declare the objects.

**STEP 6:** Call the function.

**STEP 7:** Print the result.

**STEP 8:** stop the program.

**SOURCE CODE:**

#include <iostream>

#include <cstring>

using namespace std;

class Student {

private:

char name[30];

long rollNo;

char branch[30];

public:

Student(const char \* name, long rollNo,

const char \* branch)

{

strcpy(this -> name, name);

this -> rollNo = rollNo;

strcpy(this -> branch, branch);

}

void show() {

cout << "Student Details are" << endl;

cout << "Name " << name << endl;

cout << "Roll No " << rollNo << endl;

cout << "Branch " << branch << endl;

}

};

int main() {

Student student[3] = {

Student("Ram", 1, "CSE"),

Student("Mohan", 2, "ETC"),

Student("Sohan", 3, "Mech")

};

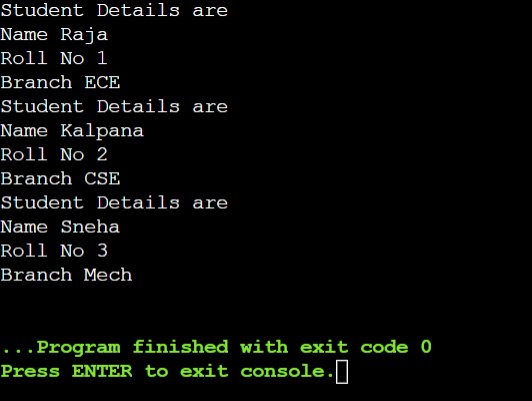
for (int i = 0; i <= 2; i++) {

student[i].show();

}

}

**OUTPUT:**



**RESULT:**

Thus the C++ program for Creating a student details using array of objects was executed successfully.

**DATE:**

**EX.NO:2c**

**PRIME NUMBER**

**AIM:**

To write a C++ program for finding whether a number is prime number or not ,using parameterized constructor.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the class named prime.

**STEP 3:** Declare the variables and initialize it.

**STEP 4:** Declare the condition inside the for and check if (i<=a).

**STEP 5:** Declare the condition that if (a % i == 0) ,print b==0 else print b==1. **STEP 6:** Declare the function called output .

**STEP 7:** Check the value of b. if (b==1) , print the statement, else print

another   statement.

**STEP 8:** In the main function get the input “a” from the user .

**STEP 9:** Then call the function to print the result.

**STEP 10:** Print the result.

**STEP 11:** Stop the program.

**SOURCE CODE:**

#include<iostream>

using namespace std;

class Prime

{

 int a, b, i;

public:

 Prime(int x)

 {

 a = x;

 b = 1;

 {

 for (i = 2; i <= a / 2; i++)  if (a % i == 0)

 {

 b = 0;

 break;

 }

 else

 {

 b = 1;

 }

 }

 }

 void output()

 {

 if (b == 1)

 cout << a << " is Prime Number.";

 else

 cout << a << " is Not Prime Numbers.";

 }

};

int main()

{

 int a;

 cout << "\nEnter the Number:";

 cin>>a;

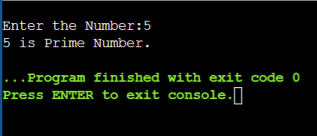
 Prime g(a);

 g.output();

 return 0;

}

**OUTPUT:**

****

**RESULT:**

Thus the C++ program for finding whether a number is prime number or not ,using parameterized constructor was executed successfully.

**DATE:**

**EX.NO:2d**

**BANK TRANSACTIONS**

**AIM:**

To write a C++ program for creating the bank transactions using copy  constructor.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare a class called bank.

**STEP 3**: Declare the variables and initialize it.

**STEP 4:** Declare the functions void deposit();,void withdraw();,void display(); inside

the   public class.

**STEP 5:** Outside the class void deposit() function , has the details about

the   deposition in the bank

**STEP 6:** Then void withdraw() function , has all the details about the withdrawal in   the

bank.

**STEP 7:** void display() function has all the details which is to be displayed.

**STEP 8:** In the main function get the input from the user acc\_no, name , acc\_type

and  balance.

**STEP 9:** By using the copy Constructor , Print the result.

**STEP 10:** stop the program.

**SOURCE CODE:**

include<iostream>

#include<stdio.h>

#include<string.h>

using namespace std;

class bank

{

 int acno;

 char nm[100], acctype[100];

 float bal;

 public:

 bank(int acc\_no, char \*name, char \*acc\_type, float balance)  {

 acno=acc\_no;

 strcpy(nm, name);

 strcpy(acctype, acc\_type);

 bal=balance;

 }

 void deposit();

 void withdraw();

 void display();

 bank( bank &obj)

 {

 obj.deposit();

 obj.withdraw();

 obj.display();

}

};

void bank::deposit()

{

 int damt1;

 cout<<"\n Enter Deposit Amount = ";  cin>>damt1;

 bal+=damt1;

}

void bank::withdraw()

{

 int wamt1;

 cout<<"\n Enter Withdraw Amount = ";  cin>>wamt1;

 if(wamt1>bal)

 cout<<"\n Cannot Withdraw Amount";  bal-=wamt1;

}

void bank::display()

{

 cout<<"\n Accout No: "<<acno;  cout<<"\n Name : "<<nm;

 cout<<"\n Account Type : "<<acctype;  cout<<"\n Balance : "<<bal;

}

int main()

{

 int acc\_no;

 char name[100], acc\_type[100];  float balance;

 cout<<"\n Enter Details: \n";

 cout<<"\n Accout No: ";

 cin>>acc\_no;

 cout<<"\n Name : ";

 cin>>name;

 cout<<"\n Account Type : ";

 cin>>acc\_type;

 cout<<"\n Balance : ";

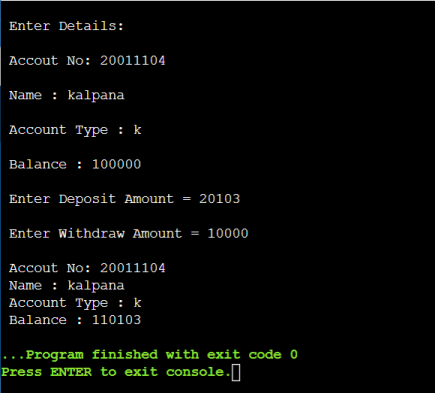
 cin>>balance;

 bank b1(acc\_no, name, acc\_type, balance);  bank b2(b1);

 return 0;

}

**OUTPUT:**

****

**RESULT:**

Thus the C++ program for creating the bank transactions using copy  constructor,

was executed successfully.

1. **FUNCTIONS IN C++**

**DATE:**

**EX.NO:3a**

**FRIEND FUNCTION**

**AIM:**

To write a C++ program for creating a class called student.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Create a class called student with a member function to get student details and

friend function to display the student details.

**STEP 3:** Get the inputs in the getDetails functions a display make the friend

function(display) to print the student details.

**STEP 4:** Call the class using an object.

**STEP 5:** Display the student details.

**STEP 6**: Print the result.

**STEP 7:** stop the program.

**SOURCE CODE:**

#include <iostream>

using namespace std;

class Student

{

char name[30];

int rollno,total;

float percentage;

public:

void getDetails()

{

cout << "Enter name: " ;

cin >> name;

cout << "Enter roll number: ";

cin >> rollno;

cout << "Enter total marks out of 500: ";

cin >> total;

percentage=(float)total/500\*100;

}

friend void display(Student);

};

void display(Student s){

cout << "Student details:"<<endl;

cout << "Name:"<< s.name<<endl;

cout << "Roll Number:" << s.rollno<<endl;

cout << "Total:" << s.total<<endl;

cout << "Percentage:" << s.percentage<<endl;

}

int main()

{

Student obj;

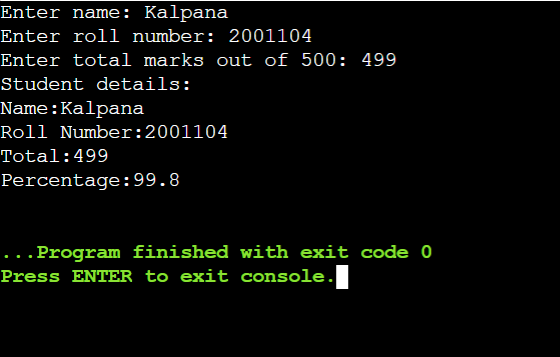
obj.getDetails();

display(obj);

return 0;

}

**OUTPUT:**



**RESULT:**

Thus the C++ program for creating a class called student was executed successfully.

**DATE:**

**EX.NO:3b**

**FUNCTION OVERLOADING**

**AIM:**

To write a C++ program for finding the area of shapes using function overloading.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Create functions with same name and different parameters.

**STEP 3:** Get inputs to pass them as parameters for the functions.

**STEP 4:** Display the result returned by the functions.

**STEP 5:** Print the result.

**STEP 6:** stop the program.

**SOURCE CODE:**

#include<iostream>

using namespace std;

int area(int side)

{

return(side \* side);

}

{

return(length \* breadth);

}

float area(float radius)

{

return(3.14 \* radius \* radius);

}

float area(float base, float height)

{

return((base \* height)/2);

}

int main()

{

int side, length, breadth;

float radius, base, height;

cout<<"Enter side of a square:";

cin>>side;

cout<<"Enter length and breadth of rectangle:";

cin>>length>>breadth;

cout<<"Enter radius of circle:";

cin>>radius;

cout<<"Enter base and height of triangle:";

cin>>base>>height;

cout<<"Area of square is"<<area(side);

cout<<"\nArea of rectangle is "<<area(length, breadth);

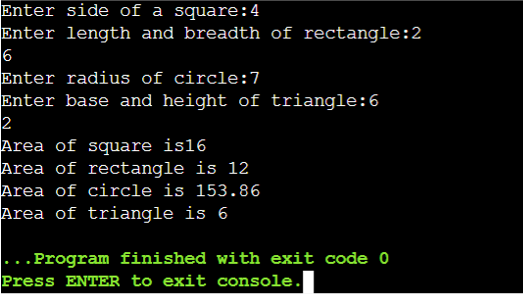
cout<<"\nArea of circle is "<<area(radius);

cout<<"\nArea of triangle is "<<area(base, height);

return 0;

}

**OUTPUT:**



**RESULT:**

Thus the C++ program finding the area of shapes using function overloading was executed successfully.

**DATE:**

**EX.NO:3c**

**SUBTRACTION AND MULTIPLICATION OF**

**TWO COMPLEX NUMBERS USING**

**OPERATOR OVERLOADING**

**AIM:**

To write a C++ program for finding the subtraction and multiplication of two

complex number using operator overloading.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the class named Complex .

**STEP 3:** Get the input from the user.

**STEP 4:** Create the function Complex operator - (Complex c2)

to overload the subtraction operation and Complex operator \*

(Complex c2) to overload multiplication operation.

**STEP 5:** Declare the function void output() to print the result.

**STEP 6:** Declare the function void print() to print the output.

**STEP 7:** In the main function create the object and override the function.

**STEP 8:** Call the function to print the result.

**STEP 9:** print the result with respect to the condition.

**STEP 10:** stop the program.

**SOURCE CODE:**

#include <iostream>

using namespace std;

class Complex

{

private:

float real;

float imag;

public:

Complex(): real(0), imag(0){ }

void input()

{

cout << "Enter the coefficients ";

cin >> real;

cin >> imag;

}

Complex operator - (Complex c2)

{

Complex temp;

temp.real = real - c2.real;

temp.imag = imag - c2.imag;

return temp;

}

Complex operator \* (Complex c2)

{

Complex temp;

temp.real=(real\*c2.real)-(imag\*c2.imag);

temp.imag=(real\*c2.imag)+(imag\*c2.real);

return temp;

}

void output()

{

if(imag < 0)

cout << "\nSubraction: "<< real << imag << "i";

else

cout << "\nSubraction: " << real << "+" << imag << "i";

}

void print()

{

if(imag < 0)

cout << "\nMultiplication: "<< real << imag << "i";

else

cout << "\nMultiplicion: " << real << "+" << imag << "i";

}

};

int main()

{

Complex c1, c2, subraction,multiplicion;

cout<<"Enter first complex number:\n";

c1.input();

cout<<"Enter second complex number:\n";

c2.input();

subraction = c1 - c2;

multiplicion = c1\*c2;

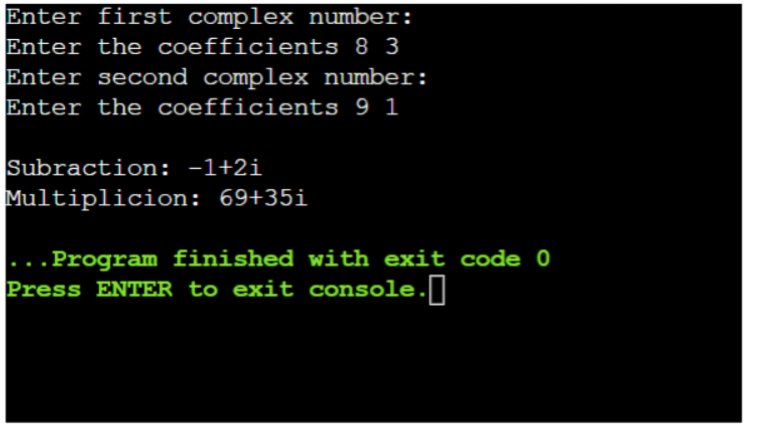
subraction.output();

multiplicion.print();

return 0;

}

**OUTPUT:**



**RESULT:**

Thus the C++ program for finding the subtraction and multiplication of two

complex number using operator overloading was executed successfully.

**DATE:**

**EX.NO:3d**

**OVERLOADING**

**AIM:**

To write a C++ program to find the subtraction and multiplication of two complex numbers using unary operation overloading.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Create a class poly with constructors with initialized variables.

**STEP 3:** Modify the + operator such that it add the like coefficients of two polynomial **STEP 4:** Modify the << operator such that it display the added coefficients in a polynomial form.

**STEP 5:** Use the modified operators to perform the operations and display the performed

operations.

**STEP 6:** print the result.

**STEP 7:** stop the program.

**SOURCE CODE:**

#include<iostream>

using namespace std;

class poly

{

    int a,b,c;

    public:

        poly()

        {

            a=0;

            b=0;

            c=0;

        }

        poly(int x, int y,int z)

        {

            a=x;

            b=y;

            c=z;

        }

        poly operator + (poly T)

        {

            poly R;

            R.a=a+T.a;

            R.b=b+T.b;

            R.c=c+T.c;

            return R;

        }

        friend ostream &operator << (ostream &out, poly &T)

        {

            out<<T.a<<"x^2 + "<<T.b<<"x +"<<T.c;

            return out;

        }

};

int main()

{

    poly q1(3,4,-2), q2(0,-4,10), sum;

    cout<<"two polynomial: \n";

    sum=q1+q2;

    cout<<q1<<":q1\n";

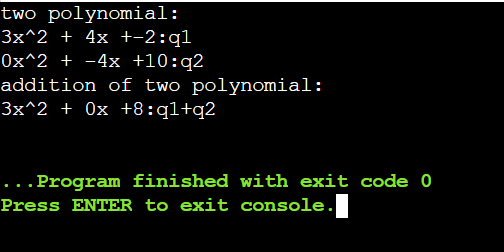
    cout<<q2<<":q2\n";

    cout<<"addition of two polynomial: \n";

    cout<<sum<<":q1+q2\n";

}

**OUTPUT:**



**RESULT:**

Thus the C++ program for finding the subtraction and multiplication of two complex numbers using unary operation overloading was executed successfully.

1. **INHERITANCE**

**TO CALCULATE THE TOTAL MARKS OF**

**EACH STUDENT**

**DATE:**

**EX.NO:4a**

**AIM:**

To write a C++ program for calculating the total marks of each student.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the class named class Marks().

**STEP 3:** Declare the variables and initialize it.

**STEP 4**: Declare the function void GenerateMarks(int num).

**STEP 5:** Declare the condition to generate the marks .

**STEP 6:** Assign the for loop and check if (i&lt;num) and execute the

condition marks[i] = rand() % 5 + 1.

**STEP 7:** Declare the another function named int sum\_of\_marks() to print

the sum of the marks.

**STEP 8:** Declare the function named double avarage() to print the

average of the marks.

**STEP 9:** Declare the functions void set\_marks(int number\_of\_marks, int

newmarks) , int get\_marks(int number\_of\_marks) , void

set\_num\_marks(int num) to return and calculate the marks.

**STEP 10:** In the main function declare the variable to get the number of

student from the user.

**STEP 11:** Declare the for loop to generate the marks and print marks.

**STEP 12:** print the result with respect to the condition.

**STEP 13:** stop the program.

**SOURCE CODE:**

#include <iostream>

#include <cmath>

using namespace std;

class Marks

{

public:

int rollNumber;

char \*name;

int \*marks;

int num\_marks;

void GenerateMarks(int num)

{

num\_marks = num;

marks = new int[num];

for (int i = 0; i < num; i++)

marks[i] = rand() % 5 + 1;

}

int sum\_of\_marks()

{

int sum = 0;

for (int i = 0; i < num\_marks; i++)

sum += marks[i];

return sum;

}

double avarage()

{

return round(((double)sum\_of\_marks() / num\_marks) \* 100) / 100;

}

void set\_marks(int number\_of\_marks, int newmarks)

{

marks[number\_of\_marks] = newmarks;

}

int get\_marks(int number\_of\_marks)

{

return marks[number\_of\_marks];

}

void set\_num\_marks(int num)

{

num\_marks = num;

delete[]marks;

marks = new int[num];

}

~Marks() {

delete[]marks;

}

};

class Physics : public Marks

{

public:

};

class Chemistry : public Marks

{

public:

};

class Mathematics : public Marks

{

public:

};

int main()

{

int num;

cout << "Enter the number of students ";

cin >> num;

Physics \*c\_phys = new Physics[num];

Chemistry \*c\_chem = new Chemistry[num];

Mathematics \*c\_math = new Mathematics[num];

for (int i = 0; i < num; i++)

{

c\_phys[i].rollNumber = c\_chem[i].rollNumber = c\_math[i].rollNumber = i + 1;

c\_phys[i].GenerateMarks(9);

c\_chem[i].GenerateMarks(5);

c\_math[i].GenerateMarks(2);

}

cout << "\nThe total marks of each student of a class in Physics, Chemistry and Mathematics:\n";

for (int i = 0; i < num; i++)

{

cout << "Roll number of student is " << c\_phys[i].rollNumber << endl;

cout << "The total marks in Physics is " << c\_phys[i].sum\_of\_marks() << endl;

cout << "The total marks in Chemistry is " << c\_chem[i].sum\_of\_marks() << endl;

cout << "The total marks in Mathematics is " << c\_math[i].sum\_of\_marks() << endl;

cout << endl;

}

cout << "\nThe average marks=\n";

for (int i = 0; i < num; i++)

{

cout << "Roll number of student is " << c\_phys[i].rollNumber << endl;

cout << "The average mark in Physics is " << c\_phys[i].avarage() << endl;

cout << "The average mark in Chemistry is " << c\_chem[i].avarage() << endl;

cout << "The average mark in Mathematics is " << c\_math[i].avarage() << endl;

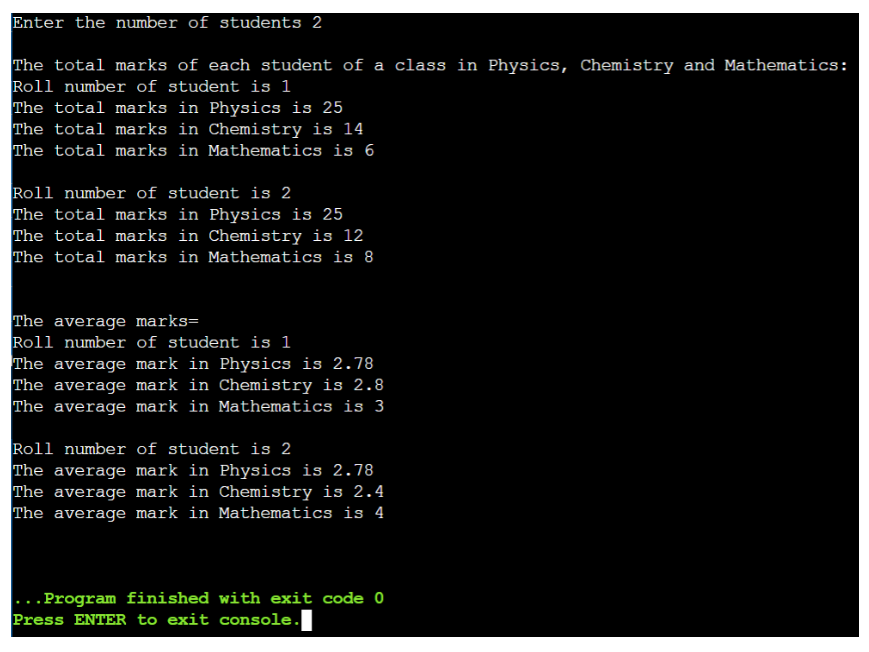
cout << endl;

}

return 0;

}

**OUTPUT:**



**RESULT:**

Thus the C++ program for calculating the total marks of each student

was executed successfully.

**DATE:**

**EX.NO:4b**

**AREA OF RECTANGLE AND CIRCLE**

**AIM:**

To write a C++ program for finding the area of rectangle and circle.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Create a class polygon and declare members functions, input() to get the

dimensions of the shape and area() to print the area of the shape.

**STEP 3:** Create two derived classes rectangle and circle to get the input and print them.

**STEP 4:** Create two objects r and c to call the classes.

**STEP 5:** Print the result.

**STEP 6:** stop the program.

**SOURCE CODE:**

#include<iostream>

using namespace std;

class Polygon

{

public:

void input();

void area();

};

class Rectangle : public Polygon

{

public:

int l,b;

void input()

{

cout<<"Enter the length and breadth of the rectangle: ";

cin>>l>>b;

}

void area()

{

int area = l \* b;

cout<<"The area of the Rectangle is "<<area<<"\n";

}

};

class Circle : public Polygon

{

public:

float r;

void input()

{

cout<<"Enter the radius of the circle: ";

cin>>r;

}

void area()

{

float area = 3.14 \* r \* r;

cout<<"The area of the Circle is "<<area<<"\n";

}

};

int main()

{

Rectangle r;

Circle c;

r.input();

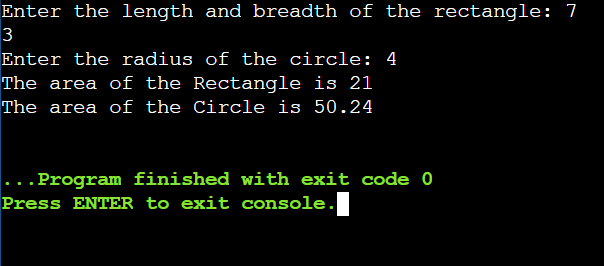
c.input();

r.area();

c.area();

}

**OUTPUT:**



**RESULT:**

Thus the C++ program for Calculating Electricity bill with else if ladder was executed successfully.

**(V) TEMPLATES**

**DATE:**

**EX.NO:5a**

**FUNCTION TEMPLATE**

**AIM:**

To write a C++ program for function template that takes a single type parameter (T) and accepts four function arguments: an array of T, a start index, a stop index (inclusive), and an optional initial value. The function returns the sum of all the array elements in the specified range and the initial value. Use the default constructor of T for the default initial value.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Declare the class template T.

**STEP 3:** Declare the condition.

**STEP 4:** In the main function declare the variables and initialize it.

**STEP 5:** Declare the condition float result=sum(arr,0,4);

**STEP 6:** Print the result with respect to the condition .

**STEP 7:** stop the program.

**SOURCE CODE:**

#include<iostream>

using namespace std;

template <class T>

T sum(T arr[], int start , int end,T init=T())

{

T total=init;

for(int i=start;i<=end;i++)

{

total+=arr[i];

}

return total;

}

int main()

{

float arr[]={1.0,2.0,3.0,5.0,6.0};

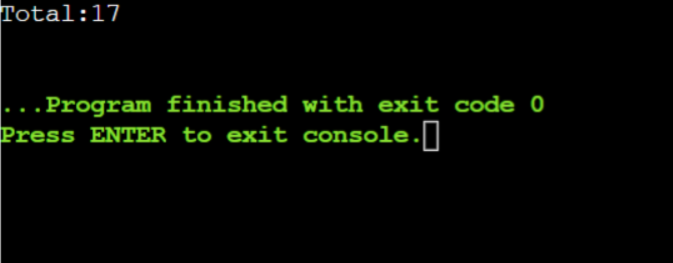
float result=sum(arr,0,4);

cout<<"Total:"<<result<<endl;

return 0;

}

**OUTPUT:**



**RESULT:**

Thus the C++ program for function template was executed successfully.

**EX.NO:5b**

**OPERATIONS OF STACK DATA STRUCTURE**

**DATE:**

**AIM:**

To write a C++ program for class template that to implement the operations of the stack data

structure.

**ALGORITHM:**

**STEP 1:** start the program.

**STEP 2:** Create the class template T.

**STEP 3:** Declare the class named stack.

**STEP 4:** Create the function named void push(T i).

**STEP 5:** Declare the condition.

**STEP 6:** In the main function ,call the function.

**STEP 7:** Print the result with respect to the condition.

**STEP 8:** stop the program.

**SOURCE CODE:**

#include <iostream>

#include <string>

using namespace std;

template <class T>

class Stack

{

public:

Stack();

void push(T i);

T pop();

private:

int top;

T st[100];

};

template <class T>

Stack<T>::Stack()

{

top = -1;

}

template <class T>

void Stack<T>::push(T i)

{

st[++top] = i;

}

template <class T>

T Stack<T>::pop()

{

return st[top--];

}

int main ()

{

Stack<int> int\_stack;

Stack<string> str\_stack;

int\_stack.push(67);

str\_stack.push("Hello");

str\_stack.push("Codezclub");

cout << int\_stack.pop() << endl;

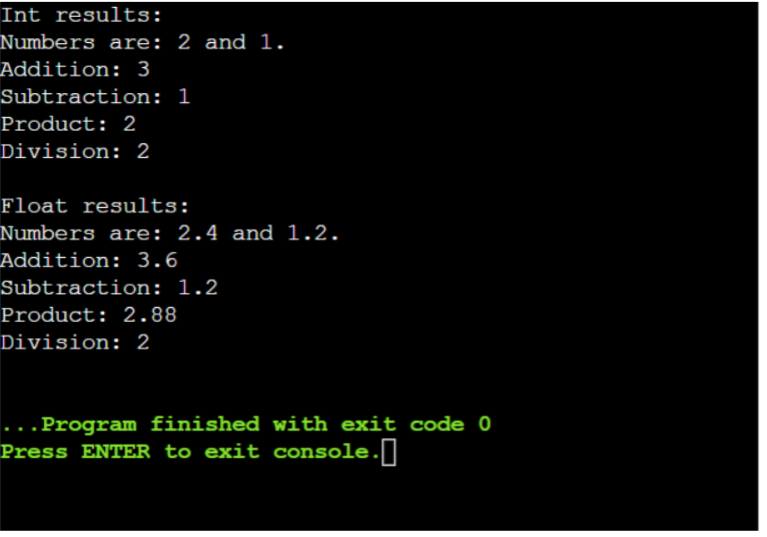
cout << str\_stack.pop() << endl;

cout << str\_stack.pop() << endl;

return 0;

}

**OUTPUT:**



**RESULT:**

Thus the C++ program for operations on stack data structure was executed successfully.