Date: **Roll No.:**

2	4	B	1	1	A	I	4	5	6

WEEK 11

Program No:11.1

Develop a C++ program that demonstrates exception handling using try, throw, and catch blocks.

Aim: To develop a C++ program that demonstrates exception handling using try, throw, and catch blocks.

Description:

Exception handling in C++ is a powerful feature that helps you deal with unexpected errors during program execution—like dividing by zero, accessing invalid memory, or failing to open a file.

- try block: Wraps the code that might cause an error.
- throw statement: Signals that an error has occurred and sends an exception.
- catch block: Receives and handles the exception, preventing the program from crashing.

This mechanism allows your program to respond gracefully to problems, display meaningful messages, and continue running or exit safely.

Syntax:

```
try {
   // Code that may cause an exception
   throw exception_value;  // Throwing an exception
}
catch (exception_type variable) {
   // Code to handle the exception
}
```

Program:

```
#include<iostream>
using namespace std;
int main()
{
cout<<"Roll no:24B11AI456"<<endl;
    int numerator,denominator;
  double result;
    cout<<"enter numerator:";
  cin>>numerator;
    cout<<"enter denominator:";
    cin>>denominator;
    try
{
```

```
    if(denominator==0)


    throw denominator;
        result=(double)numerator/denominator;
        cout<<"result ="<<result<<endl;
    }
    catch(int e)
    {
        cout<<"Error:Division by zero is not allowed!"<<endl;
    }
cout<<"end Program..."<<endl;
    return 0;


    }
```

Output 1:

Roll no:24B11AI456
enter numerator:5
enter denominator:0
Error:Division by zero is not allowed!
end Program...

Output 2:

Roll no:24B11AI456
enter numerator:2
enter denominator:5
result =0.4
end Program...

Date: **Roll No.:**

2	4	B	1	1	A	I	4	5	6

Program No :10.2

Develop a C++ program to illustrate the use of multiple catch statements, where different types of exceptions are caught and handled differently.

Aim : Develop a C++ program to illustrate the use of multiple catch statements, where different types of exceptions are caught and handled differently.

Description :

This C++ program shows how to use multiple catch blocks to handle different types of exceptions. Each catch block is designed to respond to a specific error type—like int, char*, or std::exception—so the program can react appropriately based on what went wrong. This makes error handling more precise and flexible.

Syntax :

```
try {
   // Code that may throw different types of exceptions
   throw exception_value;
}
catch (int e) {
   // Handle integer exception
}
catch (const char* msg) {
   // Handle string literal exception
}
catch (const std::exception& ex) {
   // Handle standard exception
}
```


Program:

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
        cout<<"Roll no:24B11AI456"<<endl;
        int num1,num2;
        char op;
        cout<<"Simple calculator"<<endl;
```

```
        cout<<"enter first number:";

        cin>>num1;
        cout<<"enter second number:";
        cin>>num2;
        cout<<"Enter an operator(+,-,*,/):";
        cin>>op;
try
        {
                if(op!='+'&&op!='-'&&op!='*'&&op!='/')
                throw string("invalid operator!please use +,-,*,/.");
                if(num1<0||num2<0)
                throw -1;
                if(op!='/'&&num2==0)
                throw 0;
                double result;
                switch(op)
                {
                        case '+':result=num1+num2;
                        break;
                                case '-':result=num1-num2;
                        break;
                                case '*':result=num1*num2;
                        break;
                                case '/':result=num1/num2;
                        break;
   }
                cout<<"result : "<<result<<endl;
        }
        catch(int e)
        {
                cout<<"Error:Division by zero is not allowed!"<<endl;
        }
        catch(double e)
        {
```

```
cout<<"Error:Negative numbers are not allowed!"<<endl;
        }
        catch(string e)
        {
                cout<<"Error:"<<e<<endl;
        }
        cout<<"program execution completed successfully"<<endl;
        return 0;
}
```

Output:

```
Roll no:24B11AI456
Simple calculator
enter first number:3
enter second number:5
Enter an operator(+,-,*,/):+
result : 8
program execution completed successfully
```