S.No: 1	Exp. Name: ***Write a C++ program to find the sum of individual digits of a positive integer.***	Date: 2023-02-15

Aim:

Write a C++ program to find the sum of individual digits of a positive integer.

Source Code:

sum.cpp

```
#include <iostream>
using namespace std;
int main(){
        int n,sum=0,m;
        cout << "Enter a number: ";
        cin>>n;
        while(n>0)
        {
                m=n%10;
                sum=sum+m;
                n=n/10;

        }
        cout<< "Sum is= "<<sum<<endl;
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter a number:
2563
Sum is= 16

ID: 22SCSE1180156
E2UC201C-2022-Section-23
Galgotias University Greater Noida

S.No: 2	Exp. Name: ***Write a C++ program to find Largest and Smallest of List of integers***	Date: 2023-02-19

Aim:

Write a C++ program to read an array of integers (with max size 10) and print the `largest` and the `smallest` of the given numbers.

Print the output as shown in the test cases. **Note:** Do use the **printf()** function with a **newline** character (`\n`) at the end.

Source Code:

ArraysDemo5.cpp

```
#include <iostream>
using namespace std;
int main(){
        int n;
        int arr[10];

        cout << "Enter n : ";
        cin >> n;
        cout << "Enter " <<n <<" Values : ";
        for(int i=0; i<n; i++){
                cin >> arr[i];
        }

        int largest = arr[0], smallest =arr[0];
        for(int i=1; i<n; i++){
                if(arr[i]>largest){
                        largest = arr[i];
                }
                if (arr[i]<smallest){
                        smallest= arr[i];
                }
        }

        cout<<"Largest element = "<<largest<<endl;
        cout<<"Smallest element = "<<smallest<<endl;
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter n :
4
Enter 4 Values :
24 38 15 13
Largest element = 38
Smallest element = 13

Test Case - 2
User Output
Enter n :
5
Enter 5 Values :
-1 -2 -3 -44 -33
Largest element = -1
Smallest element = -44

S.No: 3	Exp. Name: ***Write a C++ program to Overload new and delete Operators***	Date: 2023-02-25

ID: 22SCSE1180156

Aim:

Write a **C++** program to overload `new` and `delete` operators as `member functions` to allocate memory to the class and destroy it.

Note: Write a class `Student` which contains two members `name` and `id`, a `constructor` and overloaded operator functions `new` and `delete` in the below code.

The output of the program is:

```
The student name : Saraswathi
The student id : 555
```

Note: Driver code is given in `OverloadNewDelete2.cpp` tab and you have to complete the code in `OverloadNewDelete2a.cpp` tab.

Source Code:

OverloadNewDelete2.cpp

```
#include <iostream>
using namespace std;
#include "OverloadNewDelete2a.cpp"
int main() {
        Student *s;
        s = new Student("Saraswathi", 555);
        s -> display();
        delete s;
        return 0;
}
```

OverloadNewDelete2a.cpp

```
//Start writing required code to complete the functionality here
#include<iostream>
#include<cstring>
using namespace std;
class Student{
private:
 char* name;
 int id;
public:
  Student(const char* _name,int id):id(id){
name= new char[strlen(_name)+1];
strcpy(name,_name);
}
~Student(){
        delete[] name;
}
void display(){
        cout<<"The student name : "<<name<<endl;
        cout<<"The student id : "<<id<<endl;
}
void* operator new (size_t size){
        void* ptr=::operator new(size);

        return ptr;
}
void operator delete(void* ptr){
        ::operator delete(ptr);
}
};
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
The student name : Saraswathi
The student id : 555

S.No: 4	Exp. Name: ***C++ program that implement bubble sort, to sort a given list of integer in ascending order***	Date: 2023-02-26

Aim:

Write a C++ program that implement bubble sort, to sort a given list of integer in ascending order.

Source Code:

BubbleSort.cpp

```
#include <iostream>
#include <vector>
using namespace std;
void bubbleSort(vector<int>& a){
        int n=a.size();
        for (int i=0; i<n-1;i++){
                for(int j=0;j<n-i-1;j++){
                        if (a[j]>a[j+1]){
                                int temp = a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
                }
        }
}
int main(){
        int n;
        cout<<"Enter the number of data element to be sorted : ";
        cin>>n;
        vector<int> a(n);
        for(int i=0;i<n;i++){
                cout<<"Enter element "<<i+1<<": ";
                cin>>a[i];
        }
        bubbleSort(a);
        cout<<"Sorted Data ";
        for(int i=0;i<n-1;i++ ){
                cout<<a[i]<<" ";
        }
        cout<<a[n-1];
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the number of data element to be sorted :
5
Enter element 1:

45
Enter element 2:
12
Enter element 3:
89
Enter element 4:
0
Enter element 5:
1
Sorted Data 0 1 12 45 89

Test Case - 2
User Output
Enter the number of data element to be sorted :
8
Enter element 1:
100
Enter element 2:
60
Enter element 3:
0
Enter element 4:
23
Enter element 5:
2
Enter element 6:
1
Enter element 7:
78
Enter element 8:
999
Sorted Data 0 1 2 23 60 78 100 999

Test Case - 3
User Output
Enter the number of data element to be sorted :
3
Enter element 1:
10001
Enter element 2:
999
Enter element 3:
200
Sorted Data 200 999 10001

S.No: 5	Exp. Name: ***Write the code to concatenate two strings***	Date: 2023-02-26

Aim:

Write a C++ program illustrating user-defined string processing functions using pointers to concatenate two strings.

Source Code:

concatenation.cpp

```
#include <iostream>
using namespace std;
char* strct(char* str1, const char* str2){
        char* ptr=str1;
        while (*ptr){
                ptr++;
        }
        while(*str2){
                *ptr=*str2;
                ptr++;
                str2++;
        }
        *ptr='\0';
        return str1;
}
int main(){
        char str1[100],str2[100];
        cout<<"enter first string: ";
        cin.getline(str1,100);
        cout<<"enter second string: ";
        cin.getline(str2,100);
        char* result=strct(str1,str2);
        cout<<"The concatenated string is "<<result<<endl;
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
enter first string:
code
enter second string:
tantra
The concatenated string is codetantra

Test Case - 2
User Output

```
enter first string:
cpp
enter second string:
programming
The concatenated string is cppprogramming
```

S.No: 6	Exp. Name: ***Write the code to copy a string***	Date: 2023-02-26

Aim:

Write a C++ program illustrating user-defined string processing functions using pointers to copy a string.

Source Code:

copyString.cpp

```
#include <iostream>
using namespace std;
void strc(char* target, const char* source){
        while(*source){
                *target=*source;
                target++;
                source++;
        }
        *target='\0';
}
int main (){
        char source[100],target[100];
        cout<<"enter a string: ";
        cin.getline(source,100);
        strc(target,source);
        cout<<"target string: "<<target<<endl;
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
enter a string:
c++ programming
target string: c++ programming

Test Case - 2
User Output
enter a string:
CodeTantra
target string: CodeTantra

S.No: 7	Exp. Name: ***Write the code to find string length using pointers***	Date: 2023-02-26

Aim:

Write a C++ program illustrating user-defined string processing functions using pointers to find string length.

Source Code:

stringLength.cpp

```
#include<iostream>
using namespace std;
int strl(const char* str){
 const char* ptr =str;
        while (*ptr){
                ptr++;
        }
        return ptr-str;
}
int main() {
        char str[100];
        cout<<"enter the string: ";
        cin.getline(str,100);
        int len= strl(str);
        cout<<len<<endl;
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
enter the string:
stringlength
12

Test Case - 2
User Output
enter the string:
Cppprogramming
14

S.No: 8	Exp. Name: ***Write a C++ program to make a simple Calculator to Add, Subtract, Multiply or Divide using switch-case***	Date: 2023-02-26

Aim:

Write a program to read two integer values and an arithmetic operator, depending on the operator perform different arithmetic operations.

If integer values **2** and **3** are given with operator **+**, then the output should be 2 + 3 = 5.

If integer values **6** and **3** are given with operator **/**, then the output should be 6 / 3 = 2.

If other than arithmetic operator is given, then display "**Error! Operator is not correct**".

At the time of execution, the program should print the message on the console as:

```
Enter two integer values :
```

For example, if the user gives the **input** as:

```
Enter two integer values : 12 10
```

Next, the program should print the message on the console as:

```
Enter an arithmetic operator :
```

For example, if the user gives the **input** as:

```
Enter an arithmetic operator : +
```

then the program should **print** the result as:

```
12 + 10 = 22
```

Note: Do use **newline** character (\n) at the end.

Source Code:

SwitchCaseDemo3.cpp

```
#include <iostream>
using namespace std;
int main(){
        int a,b;
        char n;
        cout<<"Enter two integer values : ";
        cin>>a>>b;
        cout<<"Enter an arithmetic operator : ";
        cin>>n;
        switch (n){
          case '+':
                 cout<<a<<" + "<<b<<" = "<<a+b<<"\n";
                 break;
          case '-':
                 cout<<a<<" - "<<b<<" = "<<a-b<<"\n";
                 break;
          case '*':
                 cout<<a<<" * "<<b<<" = "<<a*b<<"\n";
                 break;
          case '/':
                 cout<<a<<" / "<<b<<" = "<<a/b<<"\n";
                 break;
          case '%':
           cout<<a<<" % "<<b<<" = "<<a%b<<"\n";
            break;
         default:
           cout<<"Error! Operator is not correct"<<endl;
        }
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter two integer values :
12 13
Enter an arithmetic operator :
+
12 + 13 = 25

Test Case - 2
User Output
Enter two integer values :
5 8
Enter an arithmetic operator :
*
5 * 8 = 40

Test Case - 3
User Output
Enter two integer values :
123 12
Enter an arithmetic operator :
%
123 % 12 = 3

Test Case - 4
User Output
Enter two integer values :
67 89
Enter an arithmetic operator :
#
Error! Operator is not correct

S.No: 9	Exp. Name: ***Program to illustrate default constructor***	Date: 2023-02-26

Aim:

Fill the below missing c++ program to calculate the area of the rectangle wall using the default constructor.

Source Code:

default.cpp

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

// declare a class
class  Wall {
  private:
    //declare the variables
    float a,b;

  public:
    // default constructor to initialize variable
    Wall() {
        cout<<"Enter the length: ";
        cin>>a;
        cout<<"Enter the breadth: ";
        cin>>b;
        cout<<"Area = "<<a*b<<endl;
    }
};

int main() {
  Wall wall1;
  return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the length:
10
Enter the breadth:
8
Area = 80

Test Case - 2
User Output
Enter the length:
26.89

```
Enter the breadth:
69.25
Area = 1862.13
```

S.No: 10	Exp. Name: ***Program to illustrate copy constructor***	Date: 2023-02-26

Aim:

Fill in the below missing C++ program to demonstrate the student details using copy constructor

Source Code:

copy.cpp

```
#include <iostream>
#include <string.h>
using namespace std;
class student {
int rn;
string name;


public:
        student(int rn,string name){
                this->rn=rn;
                this->name=name;
        };
        student(const student& other ) // copy constructor
        {
                this->rn=other.rn;
                this->name=other.name;
        }
        void display();
};

void student::display()
{
cout<<rn<<"  "<<name<<endl;
}
int main()
{
        int rno;
        string name;
        cout<<"Enter roll number: ";
        cin>>rno;
        cout<<"Enter student name: ";
        cin>>name;
        student s1(rno,name);
        student s2=s1;
        s1.display();
        s2.display();


        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter roll number:
25
Enter student name:
Karna
25 Karna
25 Karna

S.No: 11	Exp. Name: ***Program to illustrate parameterized constructor***	Date: 2023-02-26

Aim:

Fill in the below missing c++ program to calculate the area of the rectangle wall using the parameterized constructor.

Source Code:

parameterized.cpp

```
#include<iostream>
using namespace std;
class wall
{
   private:
     //declare variables...
     int length;
     int breadth;

   public:
     wall (int l,int b) //parameterized constructor to initialize l and b
     {
         length = l;
         breadth = b;
      }
      int area() //function to find area
      {
      return length*breadth;
      }
      void display() //function to display the area
      {
         cout<<"Area = "<<area()<<endl;
      }
};

int main()
{
        int l,b;
        cout<<"Enter length: ";
        cin>>l;
        cout<<"Enter breadth: ";
        cin>>b;
    wall c(l,b);  //initializing the data members of object 'c' implicitly
        //call area function
        c.area();

        //call display function
    c.display();
    return 0;
 }   //end of program
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter length:
25
Enter breadth:
40
Area = 1000

Test Case - 2
User Output
Enter length:
36
Enter breadth:
52
Area = 1872

S.No: 12	Exp. Name: ***Write a Program to Implement a Class STUDENT having the Following Members:***	Date: 2023-02-26

Aim:

Write a Program to Implement a Class STUDENT having the Following Members:

MEMBER	DESCRIPTION
Data Members	
Sname	Name of the student
Marks array	Marks of the students
Total	Total marks obtained
TMax	Total Maximum marks
Member Functions	
Assign()	Assign initial Values
Compute()	To compute total and average
Display()	To display the data

Source Code:

student.cpp

```
#include <iostream>
#include <cstring>
using namespace std;
class STUDENT{
        char Sname[30];
        int Marks[6];
        int Total=0;
        int TMax;
        public:
        void Assign(){
                cout<<"Enter Student Name: ";
                cin.getline(Sname,30);
                for(int i=0;i<6;i++){
                        cout<<"Enter marks of subject "<<i+1<<": ";
                        cin>>Marks[i];
                }
                cout<<"Enter maximum total marks: ";
                cin>>TMax;
        }
   void Compute(){
                for(int i=0;i<6;i++){
                        Total=Total+Marks[i];
                }
        }
    void Display(){
        cout<<"Student Name: "<<Sname<<endl;
        cout<<"Marks are"<<endl;
        for(int i=0;i<6;i++){
                cout<<"Subject "<<i+1<<" : "<<Marks[i]<<endl;
        }
        cout<<"Total: "<<Total<<endl;
        cout<<"Percentage: "<<(float)Total/6<<endl;
    }
};
int main(){
        STUDENT s1;
        s1.Assign();
        s1.Compute();
        s1.Display();
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter Student Name:
Arjun
Enter marks of subject 1:
95
Enter marks of subject 2:
26

Enter marks of subject 3:
53
Enter marks of subject 4:
84
Enter marks of subject 5:
95
Enter marks of subject 6:
74
Enter maximum total marks:
600
Student Name: Arjun
Marks are
Subject 1 : 95
Subject 2 : 26
Subject 3 : 53
Subject 4 : 84
Subject 5 : 95
Subject 6 : 74
Total: 427
Percentage: 71.1667

S.No: 13	Exp. Name: ***Write a C++ program to Subtract two Complex numbers by Overloading - operator***	Date: 2023-03-11

Aim:

Write a C++ program to overload the binary - operator as a member function to subtract two complex numbers.

Source Code:

binaryMinus.cpp

```
#include<iostream>
using namespace std;
class complex {
        public:
        double real, imag;
        complex operator - (complex c){
                return{real-c.real, imag-c.imag};

    }
};
int main(){
        complex c1,c2,c3;
        cout<<"Enter real and imaginary parts : ";
        cin>>c1.real>>c1.imag;
        cout<<"Enter real and imaginary parts : ";
        cin>>c2.real>>c2.imag;
        c3=c1-c2;
        cout<<"Subtraction of two complex numbers : "<<c3.real<<" + i"<<c3.imag<<endl;
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter real and imaginary parts :
8 5
Enter real and imaginary parts :
3 6
Subtraction of two complex numbers : 5 + i-1

Test Case - 2
User Output
Enter real and imaginary parts :
8 -9
Enter real and imaginary parts :
5 -2
Subtraction of two complex numbers : 3 + i-7

S.No: 14	Exp. Name: ***Write a C++ program to find Addition of different Data type values using Function Overloading***	Date: 2023-03-12

Aim:

Write a **C++** program to find the `addition` of two integer values, two float values and two character values using **function overloading**.

At the time of execution, the program should print the following messages one by one on the console as:

```
Enter two integer values :
Enter two float values :
Enter two char values :
```

For example, if the user gives the **input** as:

```
Enter two integer values : 54 3
Enter two float values : 67.89 23.456
Enter two char values : A 9
```

then the program should **print** the result as:

```
Sum of two integers : 57
Sum of two floats : 91.346
Sum of two characters : z
```

Note: Do use the **cout** with a **newline** character to display the output.

Source Code:

FunctionOverloading2.cpp

```
#include <iostream>
using namespace std;
#include "Add.h"

int main() {
        int a, b;
        float c, d;
        char p, q;
        cout << "Enter two integer values : ";
        cin >> a >> b;
        cout << "Enter two float values : ";
        cin >> c >> d;
        cout << "Enter two char values : ";
        cin >> p >> q;
        cout << "Sum of two integers : " << add(a, b) << endl;
        cout << "Sum of two floats : " << add(c, d) << endl;
        cout << "Sum of two characters : " << add(p, q) << endl;
}
```

Add.h

```
int add(int a, int b) {
            return a + b;
}

float add(float a, float b) {
            return a + b;
}

char add(char a, char b) {
            int sum = (a - '0') + (b - '0');
                return sum + 'a' - 1;  // or sum + '0' + '0' for uppercase 'U'
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter two integer values :
23 56
Enter two float values :
3.45 7.18
Enter two char values :
A 4
Sum of two integers : 79
Sum of two floats : 10.63
Sum of two characters : u

S.No: 15	Exp. Name: ***Write a C++ program to illustrate Function Overloading using inline and default arguments***	Date: 2023-03-12

Aim:

Write a program illustrate `function overloading`.

Write two overloading functions for `power(number, pwr)` where `number` is of **int** argument or **double** argument and `pwr` is an **int** argument .

Let us consider the **default argument** value for `pwr` is **2**.

At the time of execution, the program should print the following messages one by one on the console as:

```
Enter any integer value :
Enter any double value :
Enter the power value :
```

For example, if the user gives the **input** as:

```
Enter any integer value : 3
Enter any double value : 2.5
Enter the power value : 5
```

then the program should **print** the result as:

```
The square of 3 : 9
The cube of 3 : 27
The 3 to the power of 5 : 243
The square of 2.5 : 6.25
The cube of 2.5 : 15.625
The 2.5 to the power of 5 : 97.6562
```

Note: Do use the **cout** with a **newline** character to display the output.

Source Code:

FunOverloading.cpp

```
#include <iostream>
#include <cmath>

using namespace std;

int power(int num, int pwr = 2) {
            return pow(num, pwr);
}

double power(double num, int pwr = 2) {
            return pow(num, pwr);
}

int main() {
            int intNum, intPwr;
             double doubleNum;
             cout << "Enter any integer value : ";
             cin >> intNum;
             cout << "Enter any double value : ";
             cin >> doubleNum;
             cout << "Enter the power value : ";
             cin >> intPwr;
             cout << "The square of " << intNum << " : " << power(intNum) << endl;
             cout << "The cube of " << intNum << " : " << power(intNum, 3) << endl;
             cout << "The " << intNum << " to the power of " << intPwr << " : " <<
power(intNum, intPwr)<<endl;
             cout << "The square of " << doubleNum << " : " << power(doubleNum)<<endl;
             cout << "The cube of " << doubleNum << " : " << power(doubleNum, 3)<<endl;
             cout << "The " << doubleNum << " to the power of " << intPwr << " : " <<
power(doubleNum, intPwr)<<endl;
return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter any integer value :
11
Enter any double value :
2.5678
Enter the power value :
8
The square of 11 : 121
The cube of 11 : 1331
The 11 to the power of 8 : 214358881
The square of 2.5678 : 6.5936
The cube of 2.5678 : 16.931
The 2.5678 to the power of 8 : 1890.12

S.No: 16	Exp. Name: ***Write a C++ program to illustrate Friend Functions***	Date: 2023-03-12

Aim:

Write a **C++** program to **exchange** two private data members of different classes.

At the time of execution, the program should print the message on the console as:

```
Enter first value :
```

For example, if the user gives the **input** as:

```
Enter first value : 10
```

Next, the program should print the message on the console as:

```
Enter second value :
```

If the user gives the **input** as:

```
Enter second value : 20
```

then the program should **print** the result as:

```
Before swap the values are : 10 20
After swap the values are : 20 10
```

Write the classes and friend functions in `exchange.h` file.

Source Code:

FriendFunctions3.cpp

```
#include <iostream>
using namespace std;
#include "exchange.h"
int main() {
        Sample s;
        Test t;
        s.getData();
        t.getData();
        cout << "Before swap the values are : ";
        display(s, t);
        swap(s, t);
        cout << "After swap the values are : ";
        display(s, t);
}
```

exchange.h

```
#include <iostream>
using namespace std;
class Test;
class Sample{
private:
 int data;
public:
  void getData(){
        cout<<"Enter first value : ";
        cin>>data;
  }
  friend void swap(Sample& s, Test& t);
  friend void display(Sample& s, Test& t);
};
class Test{
private:
 int data;
 public:
 void getData(){
        cout<<"Enter second value : ";
        cin>>data;
 }
 friend void swap(Sample& s, Test& t);
 friend void display(Sample& s, Test& t);
};
void swap(Sample& s, Test& t){
        int temp=s.data;
        s.data=t.data;
        t.data=temp;
}
void display(Sample& s, Test& t){
        cout<<s.data<<" "<<t.data<<endl;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter first value :
99
Enter second value :
98
Before swap the values are : 99 98
After swap the values are : 98 99

S.No: 17	Exp. Name: ***Write a Program to Demonstrate Friend Class.***	Date: 2023-03-12

Aim:

Your task is to Create:

- The class "ONE" contains two member variables: "private_variable" and "protected_variable".
- The class "TWO" is declared as a friend class of "ONE". This allows the members of class "TWO" to access the private and protected members of class "ONE".
- In the main function, objects of classes "ONE" and "TWO" are created and the "display" function of class "TWO" is called and passed an object of class "ONE".
- The "display" function then displays the values of the private and protected member variables of the object of class "ONE".

Source Code:

friendclass.cpp

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

class ONE {
private:
int var1;
protected:
        int var2;

public:
        ONE(int x,int y):var1(x),var2(y){

        }
        friend class TWO;
};

class TWO {
public:
 void display(ONE obj){
        cout<<"The value of Private Variable = "<<obj.var1<<endl;
        cout<<"The value of Protected Variable = "<<obj.var2<<endl;
 }

};
// Driver code
int main()
{
        int n1,n2;
        cin>>n1>>n2;
        ONE obj1(n1,n2);
        TWO obj2;
        obj2.display(obj1);
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
10
20
The value of Private Variable = 10
The value of Protected Variable = 20

Test Case - 2
User Output
50
60
The value of Private Variable = 50
The value of Protected Variable = 60

S.No: 18	Exp. Name: ***Write a Program to Access Members of a STUDENT Class Using Pointer to Object Members.***	Date: 2023-03-12

Aim:

Write a Program to Access Members of a STUDENT Class Using Pointer to Object Members.

Source Code:

student_class.cpp

```
#include <iostream>
using namespace std;
class STUDENT{
        public:
        string name;
        int age;
        float grade;
        STUDENT(){}
        STUDENT(string x,int y, float z){
                name=x;
                age=y;
                grade=z;
        }
        void getData(){
                cin>>name>>age>>grade;
        }
        void display(){
                cout<<"Name: "<<name<<endl;
                cout<<"Age: "<<age<<endl;
                cout<<"Grade: "<<grade<<endl;
        }
};
int main(){
        STUDENT *s1=new STUDENT;
        s1->getData();
        s1->display();
        return 0;

}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
JohnDoe
22
85.4
Name: JohnDoe
Age: 22
Grade: 85.4

Test Case - 2
User Output
george
22
68.9
Name: george
Age: 22
Grade: 68.9

S.No: 19	Exp. Name: ***Write a Program to Generate Fibonacci Series and use Constructor to Initialize the Data Members.***	Date: 2023-03-12

Aim:

Write a Program to Generate Fibonacci Series and use Constructor to Initialize the Data Members.

Source Code:

fibanocci.cpp

```
#include <iostream>
using namespace std;
class Fibonacci{
private:
int num_terms;
int t1,t2;

public:
 Fibonacci(int n){
        num_terms=n;
        t1=0;
        t2=1;
 }
 void gs(){
        int next_term;
        cout<<t1<<" "<<t2<<" ";
        for(int i=2;i<num_terms;i++){
                next_term=t1+t2;
                cout<<next_term<<" ";
                t1=t2;
                t2=next_term;
        }
 }
};
int main(){
        int n;
        cout<<"No.of terms: ";
        cin>>n;
        Fibonacci fib(n);
        fib.gs();
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
No.of terms:
6

0 1 1 2 3 5

Test Case - 2
User Output
No.of terms:
8
0 1 1 2 3 5 8 13

S.No: 20	Exp. Name: ***Example on catch all exceptions***	Date: 2023-03-13

Aim:

The below program is an example for **catch-all exceptions**. Write the missing code in the below program, follow the instructions given in the comment lines.

Source Code:

exceptionhandling.cpp

```
#include<iostream>
using namespace std;
int main() {
        try {
                int a, b;
                cout << "Enter two integer values: ";
                cin >> a >> b;
      if(b==0){
        throw 0;
      }
      else{
        cout<<a/b<<endl;
      }
                /*catch(...) {
                        throw; //rethrowing the exception
                }
        }
        catch(int) {
                cout << "Second value cannot be zero"<< endl;
        }
        return 0;
}*/
}
catch (int){
        cout<<"Second value cannot be zero"<<endl;
}
catch(...){
        throw;
}
return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter two integer values:
36 3
12

Test Case - 2

User Output
Enter two integer values:
3 0
Second value cannot be zero

S.No: 21	Exp. Name: ***Write a C++ program to handle an ArithmeticException - divided by zero***	Date: 2023-03-12

Aim:

Write a **C++** program to handle an `ArithmeticException` **divide by zero** using exception handling.

In **main()** method read two integers and write code to divide the first argument by the second (as integers) and print the result (i.e the quotient).

At the time of execution, the program should print the following messages one by one on the console as:

```
Enter numerator value :
Enter denominator value :
```

For example, if the user gives the **input** as:

```
Enter numerator value : 55
Enter denominator value : 0
```

then the program should **print** the result as:

```
Exception caught : divide by zero occurred
```

If the inputs given are **"36", "4"**, then the program should print the output as:

```
Result : 9
```

Source Code:

ArithmeticException1.cpp

```
#include <iostream>
using namespace std;
int main(){
        int n,d,r;
        cout<<"Enter numerator value : ";
        cin>>n;
        cout<<"Enter denominator value : ";
        cin>>d;
        try {
                if (d==0){
                        throw "divide by zero occurred";
                }
                r=n/d;
                cout<<"Result : "<<r<<endl;
        }
        catch(const char* msg){
                cout <<"Exception caught : "<<msg<<endl;
        }
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output

```
Enter numerator value :
23
Enter denominator value :
1
Result : 23
```

Test Case - 2

User Output

```
Enter numerator value :
36
Enter denominator value :
4
Result : 9
```

Test Case - 3

User Output

```
Enter numerator value :
55
Enter denominator value :
0
Exception caught : divide by zero occurred
```

Test Case - 4

User Output

```
Enter numerator value :
23
Enter denominator value :
0
Exception caught : divide by zero occurred
```

S.No: 22	Exp. Name: ***Write a C++ program to find Subject Totals and Average Marks of a Student using Multiple Inheritance***	Date: 2023-03-13

Aim:

Write a **C++** program to

- define a base class `Internals` contains internal marks of **3** subjects
- define another base class `Externals` contains external marks of **3** subjects
- define a class `Result` derived from `Internals` and `Externals`, which finds **3 subject totals**, **total** marks and **average** marks.

At the time of execution, the program should print the message on the console as:

```
Enter internal marks of 3 subjects :
```

For example, if the user gives the **input** as:

```
Enter internal marks of 3 subjects : 30 30 29.5
```

Next, the program should print the message on the console as:

```
Enter external marks of 3 subjects :
```

If the user gives the **input** as:

```
Enter external marks of 3 subjects : 69 69 65.5
```

then the program should **print** the result as:

```
Three subject totals : 99 99 95
Total marks : 293
Average marks : 97.6667
```

Write code in `MultipleInheritance2a.cpp` and `MultipleInheritance2b.cpp` files. Partial/driver code is available in `MultipleInheritance2.cpp` file

Source Code:

`MultipleInheritance2.cpp`

```
#include <iostream>
using namespace std;
#include "MultipleInheritance2a.cpp"
class Result : public Internals, public Externals {
        private:
                float s1, s2, s3, tot, avg;
        public:
                void displayTotAvg();
};
#include "MultipleInheritance2b.cpp"
int main() {
        Result r;
        r.readInternals();
        r.readExternals();
        r.displayTotAvg();
        return 0;
}
```

MultipleInheritance2a.cpp

```
// Implement class Internals and externals as required
class Internals {
public:
   float m1,m2,m3;
    void readInternals(){
        cout<<"Enter internal marks of 3 subjects : ";
        cin>>m1>>m2>>m3;

 }
};

class Externals {
public:
    float e1,e2,e3;
    void readExternals(){
           cout<<"Enter external marks of 3 subjects : ";
        cin>>e1>>e2>>e3;
        }
};
```

MultipleInheritance2b.cpp

```
// Implement displayToAvg function of Result class here
void Result:: displayTotAvg(){
        s1=m1+e1;
        s2=m2+e2;
        s3=m3+e3;
        cout<<"Three subject totals : "<<s1<<" "<<s2<<" "<<s3<<endl;
        tot=s1+s2+s3;
        cout<<"Total marks : "<<tot<<endl;
        avg=tot/3.0;
        cout<<"Average marks : "<<avg<<endl;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter internal marks of 3 subjects :
25.75 24.5 29.5
Enter external marks of 3 subjects :
67 61 54.75
Three subject totals : 92.75 85.5 84.25
Total marks : 262.5
Average marks : 87.5

S.No: 23	Exp. Name: ***Write a C++ program on find Square and Cube of a given number using Hierarchical Inheritance***	Date: 2023-03-15

Aim:

Write a C++ program to

- **define a base class `Number` contains an integer number**
- **define a derived class `Square` derived from `Number` to find square of a given number**
- **define another derived class `Cube` derived from `Number` to find cube of a given number**

At the time of execution, the program should print the message on the console as:

```
Enter an integer number :
```

For example, if the user gives the input as:

```
Enter an integer number : 5
```

then the program should print the result as:

```
The square of 5 is : 25
```

Next, the program should print the message on the console as:

```
Enter an integer number :
```

If the user gives the input as:

```
Enter an integer number : 6
```

then the program should print the result as:

```
The cube of 6 is : 216
```

Write your code in `HierarchicalInheritance2a.cpp` and `HierarchicalInheritance2b.cpp` files.

Note: Partial code is available in `HierarchicalInheritance2.cpp` main file.

Source Code:

`HierarchicalInheritance2.cpp`

```
// Uneditable driver code
#include <iostream>
using namespace std;
#include "HierarchicalInheritance2a.cpp"
class Square : public Number {
        public:
                int getSquare();
};
class Cube : public Number {
        public:
                int getCube();
};
#include "HierarchicalInheritance2b.cpp"
int main() {
        Square s;
        s.readNumber();
        cout << "The square of " << s.getNumber() << " is : " << s.getSquare() << endl;
        Cube c;
        c.readNumber();
        cout << "The cube of " << c.getNumber() << " is : " << c.getCube() << endl;
        return 0;
}
```

HierarchicalInheritance2a.cpp

```
// Write your code here for Number class
class Number{
        protected:
          int x;
        public:
          void readNumber(){
                cout<<"Enter an integer number : ";
                cin>>x;
          }
          int getNumber(){
                return x;
          }
};
```

HierarchicalInheritance2b.cpp

```
// Implement members functionality of Square and Cube class
int Square:: getSquare(){
        return x*x;
}
int Cube :: getCube(){
        return x*x*x;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1

User Output
Enter an integer number :
5
The square of 5 is : 25
Enter an integer number :
6
The cube of 6 is : 216

Test Case - 2
User Output
Enter an integer number :
11
The square of 11 is : 121
Enter an integer number :
11
The cube of 11 is : 1331

S.No: 24	Exp. Name: ***Write a C++ program to find Total, Average and Grade of a Student using Multilevel Inheritance***	Date: 2023-03-18

Aim:

Write a **C++** program to

- define a base class `Student` contains student **id** and **name**
- define a class `Test` derived from `Student`, contains marks of **3** subjects
- define a class `Result` derived from `Test`, which finds **total**, **average** and **grade** of a Test.

The **grades** of the Test are:

- >= 75% - **Distinction**
- >= 60% and < 75% - **First class**
- >= 50% and < 60% - **Second class**
- >= 35% and < 50% - **Third class**
- < 35% - **Very poor in studie<**

At the time of execution, the program should print the message on the console as:

```
Enter student id and name :
```

For example, if the user gives the **input** as:

```
Enter student id and name : 101 Govinda
```

Next, the program should print the message on the console as:

```
Enter three subjects marks :
```

If the user gives the **input** as:

```
Enter three subjects marks : 23 31 36
```

then the program should **print** the result as:

```
Id : 101
Name : Govinda
Three subjects marks : 23 31 36
Total marks : 90
Average marks : 30
Very poor in studies
```

Write the required code in `MultilevelInheritance2a.cpp` file. Partial code is available in `MultilevelInheritance2.cpp` file.

Source Code:

MultilevelInheritance2.cpp

```
// Uneditable driver code
#include <iostream>
using namespace std;
class Student {
        private:
                int id;
                char name[30];
        public:
                void readData();
                void displayData();
};
class Test : public Student {
        protected:
                float m1, m2, m3;
        public:
                void readMarks();
};
class Result : public Test {
        private:
                float tot, avg;
        public:
                void displayTotAvgGrades();
};
#include "MultilevelInheritance2a.cpp"
int main() {
        Result r;
        r.readData();
        r.readMarks();
        r.displayData();
        r.displayTotAvgGrades();
        return 0;
}
```

MultilevelInheritance2a.cpp

```
// Type your code in this file to complete the required functionality
void Student :: readData(){
        cout<<"Enter student id and name : ";
        cin>>id>>name;
}
void Student :: displayData(){
        cout<<"Id : "<<id<<endl;
        cout<<"Name : "<<name<<endl;
}
void Test :: readMarks(){
        cout<<"Enter three subjects marks : ";
        cin>>m1>>m2>>m3;
}
void Result :: displayTotAvgGrades()
{
        cout<<"Three subjects marks : "<<m1<<" "<<m2<<" "<<m3<<endl;
        float tot=m1+m2+m3;
        cout<<"Total marks : "<<tot<<endl;
        float avg= tot/3.0;
        cout<<"Average marks : "<<avg<<endl;

        if(avg>=75)
        {
                cout<<"Distinction"<<endl;
        }
        else if(avg>=60)
        {
                cout<<"First class"<<endl;
        }
        else if(avg>=50)
        {
                cout<<"Second class"<<endl;
        }
        else if(avg>=35)
        {
                cout<<"Third class"<<endl;
        }
        else
        {
                cout<<"Very poor in studies"<<endl;
        }
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter student id and name :
101 Govinda
Enter three subjects marks :
23 31 36
Id : 101

Name : Govinda
Three subjects marks : 23 31 36
Total marks : 90
Average marks : 30
Very poor in studies

Test Case - 2
User Output
Enter student id and name :
104 Vishnu
Enter three subjects marks :
78 98 85
Id : 104
Name : Vishnu
Three subjects marks : 78 98 85
Total marks : 261
Average marks : 87
Distinction

S.No: 25	Exp. Name: ***Program on Single Inheritance***	Date: 2023-03-13

Aim:

Write a C++ program to define a base class `Student` contains student id and name, a derived class `Marks` contains marks of 3 subjects, finally find a `total` and `average` marks of a student.

At the time of execution, the program should print the message on the console as:

```
Enter student id and name :
```

For example, if the user gives the input as:

```
Enter student id and name : 2 Gayle
```

Next, the program should print the message on the console as:

```
Enter three subjects marks :
```

If the user gives the input as:

```
Enter three subjects marks : 76.56 87.63 93.45
```

then the program should print the result as:

```
Id : 2
Name : Gayle
Three subjects marks : 76.56 87.63 93.45
Total marks : 257.64
Average marks : 85.88
```

Type your code in `SingleInheritance2a.cpp` tab(file).Partial code is available in `SingleInheritance2.cpp` tab(file).

Source Code:

SingleInheritance2.cpp

```
// Uneditable driver code
#include <iostream>
using namespace std;
class Student {
        private:
                int id;
                char name[30];
        public:
                void readData();
                void displayData();
};
class Marks : public Student {
        private:
                float m1, m2, m3, total, avg;
        public:
                void readMarks();
                void displayTotAvgMarks();
};
#include "SingleInheritance2a.cpp"
```

SingleInheritance2a.cpp

```
// Type your code here to complete the functionality
void Student :: readData(){
        cout<<"Enter student id and name : ";
        cin>>id>>name;

}
void Student :: displayData(){
        cout<<"Id : "<<id<<endl;
        cout<<"Name : "<<name<<endl;
}
void Marks :: readMarks(){
        cout<<"Enter three subjects marks : ";
        cin>>m1>>m2>>m3;

}
void Marks::displayTotAvgMarks(){
        cout<<"Three subjects marks : "<<m1<<" "<<m2<<" "<<m3<<endl;
        total=m1+m2+m3;
        cout<<"Total marks : "<<total<<endl;
        avg=total/3.0;
        cout<<"Average marks : "<<avg<<endl;
}
int main(){
        Student s1;
        Marks m1;
        s1.readData();
        m1.readMarks();
        s1.displayData();
        m1.displayTotAvgMarks();
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter student id and name :
001 Smith
Enter three subjects marks :
56 76 48
Id : 1
Name : Smith
Three subjects marks : 56 76 48
Total marks : 180
Average marks : 60

Test Case - 2
User Output
Enter student id and name :
2 Gayle
Enter three subjects marks :
76.56 87.63 93.45
Id : 2
Name : Gayle
Three subjects marks : 76.56 87.63 93.45
Total marks : 257.64
Average marks : 85.88

S.No: 26	Exp. Name: ***Develop a C++ program to find the area of a rectangle by converting the member of a class square which is a friend class of rectangles.***	Date: 2023-03-18

Aim:

Use friend class: Develop a C++ program to find the area of a rectangle by converting the member of a class square which is a friend class of rectangles. Declare Rectangle as a friend of Square so that Rectangle member functions could have access to the private member of the square.

Source Code:

rectangle.cpp

```
#include <iostream>
using namespace std;
class Rectangle;
class square{
int side;
 public:
  square(int s):side(s){}
  friend class Rectangle;
};
class Rectangle{
        int l,w;
        public:
         Rectangle(square s){
                l=s.side;
                w=s.side;
         }
  void area(){
        int a=l*w;
        cout<<"Area of rectangle: "<<a<<endl;
  }
};
int main(){
        int n;
        cin>>n;
        square s1(n);
        Rectangle r1(s1);
        r1.area();
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
2
Area of rectangle: 4

Test Case - 2
User Output
10
Area of rectangle: 100

S.No: 27	Exp. Name: ***Write a C++ program to illustrate Class templates with Multiple parameters***	Date: 2023-03-28

Aim:

Write a C++ program to illustrate class templates with multiple parameters.

Write a class `MySequence` in the below program at `CtWithMultiArgs2a.cpp` which contains

- **a method setMember() used to set the array elements depending on the given subscript**
- **another method getMember() used to get the particular member from the given array.**
- **another method showElements() used to display all the elements of the given array.**

Note: Driver code is available in `CtWithMultiArgs2.cpp` file.

Source Code:

`CtWithMultiArgs2.cpp`

```
// Uneditable driver code
#include <iostream>
using namespace std;
#include "CtWithMultiArgs2a.cpp"
int main() {
        MySequence <int, 5> myInts;
        MySequence <float, 4> myFloats;
        myInts.setMember(0, 10);
        myInts.setMember(1, 20);
        myInts.setMember(2, 30);
        myInts.setMember(3, 40);
        myInts.setMember(4, 50);
        cout << "Values in myInts : ";
        myInts.showElements(5);
        myFloats.setMember(0, 1.5);
        myFloats.setMember(1, 2.5);
        myFloats.setMember(2, 3.5);
        cout << "Values in myFloats : ";
        myFloats.showElements(3);
        cout << "Value at position - 1 of myInts : " << myInts.getMember(1) << endl;
        cout << "Value at position - 4 of myInts : " << myInts.getMember(4) << endl;
        cout << "Value at position - 0 of myFloats : " << myFloats.getMember(0) << endl;
        cout << "Value at position - 2 of myFloats : " << myFloats.getMember(2) << endl;
        return 0;
}
```

`CtWithMultiArgs2a.cpp`

```
// Write your code here to meet the functional rquirements
// of the problem
template<typename T,int size>
class MySequence{
 private:
  T arr[size];
 public:
   void setMember(int index, T value)
   {
        arr[index]=value;
   }

   T getMember(int index){
        return arr[index];
   }
   void showElements(int count){
        for(int i=0; i<count; i++){
                cout<<arr[i]<<" ";
        }
        cout<<endl;
   }
};
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Values in myInts : 10 20 30 40 50
Values in myFloats : 1.5 2.5 3.5
Value at position - 1 of myInts : 20
Value at position - 4 of myInts : 50
Value at position - 0 of myFloats : 1.5
Value at position - 2 of myFloats : 3.5

S.No: 28	Exp. Name: ***Write a C++ program to illustrate the Member function templates concept***	Date: 2023-03-28

ID: 22SCSE1180156

Aim:

Write a C++ program to illustrate the member function templates concept.

Write the class `Numeric` in the below program at `MemFunTemplates2a.cpp` which contains

- **Two** private data members
- A parameterized **constructor** used to initialize two data members
- Methods **add()**, **subtract()**, **multiply()** and **division()** are used to find the addition, subtraction, multiplication and division values of two data members.

Note: Driver code is available in `MemFunTemplates2.cpp` file.

Source Code:

MemFunTemplates2.cpp

```
// Uneditable driver code
#include <iostream>
using namespace std;
#include "MemFunTemplates2a.cpp"
int main() {
        int num1, num2;
        float val1, val2;
        cout << "Enter 2 integer values : ";
        cin >> num1 >> num2;
        cout << "Enter 2 float values : ";
        cin >> val1 >> val2;
        Numeric<int> ob1(num1, num2);
        Numeric<float> ob2(val1, val2);
        cout << "Addition of 2 ints : " << ob1.add() << endl;
        cout << "Subtraction of 2 ints : " << ob1.subtract() << endl;
        cout << "Multiplication of 2 ints : " << ob1.multiply() << endl;
        cout << "Division of 2 ints : " << ob1.division() << endl;
        cout << "Addition of 2 floats : " << ob2.add() << endl;
        cout << "Subtraction of 2 floats : " << ob2.subtract() << endl;
        cout << "Multiplication of 2 floats : " << ob2.multiply() << endl;
        cout << "Division of 2 floats : " << ob2.division() << endl;
        return 0;
}
```

MemFunTemplates2a.cpp

```
// Write your code here to complete the functional requirements
// of the problem
template<class T>
class Numeric{
private:
 T num1,num2;
public:
  Numeric(T n1,T n2){
        num1=n1;
        num2=n2;
  }
  T add(){
        return num1+num2;
  }
  T subtract(){
        return num1-num2;
  }
  T multiply(){
        return num1*num2;
  }
  T division(){
        return num1/num2;
  }
};
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter 2 integer values :
55 66
Enter 2 float values :
4.7 6.8
Addition of 2 ints : 121
Subtraction of 2 ints : -11
Multiplication of 2 ints : 3630
Division of 2 ints : 0
Addition of 2 floats : 11.5
Subtraction of 2 floats : -2.1
Multiplication of 2 floats : 31.96
Division of 2 floats : 0.691176

S.No: 29	Exp. Name: ***Write a program to count the number of words and characters in a file.***	Date: 2023-03-28

Aim:

Write a program to count the number of words and characters in a file.

Source Code:

filesex1.cpp

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main() {
        string filename;
        cout << "File name: ";
        cin >> filename;

        ifstream file(filename);
        if (!file.is_open()) {
                cout << "Error: Unable to open file.\n";
                return 0;
                }
                int word_count = 0, char_count = -1;
                string word;
                while (file >> word) {
                        word_count++;
                        char_count += word.size();
                        char_count++;
                        }
                        file.close();

                        cout << "Word count: " << word_count << endl;
                        cout << "Character count: " << char_count << endl;
                        return 0;
                        }
```

file2.txt

```
The main aim of OOP is to bind together the data and the functions that operate on them so
that no other part of the code can access this data except that function
```

file1.txt

```
A Class is a user defined datatype which has data members and member functions.
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
File name:
file1.txt
Word count: 14
Character count: 79

S.No: 31	Exp. Name: ***Write a C program to Merge two Files and stores their contents in another File***	Date: 2023-03-28

Aim:

Nita is a data manager in a school and she manages the data work. She stores the data in the files. But one day she was absent and another person of her team managed the work and he stored the content in a new file.Now when Nita comes back to the work she saw that her team member stored the new data in a new file. Now she wants to store these two files in another file.
Your task is to write a program to store the data of that two files into another new file.

Source Code:

nitaFiles.cpp

ID: 22SCSE1180156

Galgotias University Greater Noida

```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

int main() {
        string file1, file2, mergedFile;
        cout << "Enter the data to be inserted in file 1: ";
        getline(cin, file1);
        cout << "Enter the data to be inserted in file 2: ";
        getline(cin, file2);

        // Writing to file1
        ofstream f1("file1.txt");
        if (f1.is_open()) {
                f1 << file1;
                f1.close();
        }
        else {
                return 1;
        }

        // Writing to file2
        ofstream f2("file2.txt");
        if (f2.is_open()) {
                f2 << file2;
                f2.close();
        }
        else {
                return 1;
        }
        // Merging files
        cout << "Merging files..." << endl;
        mergedFile = "mergedFile.txt";
        ifstream f1Read("file1.txt");
        ifstream f2Read("file2.txt");
        ofstream mergedFileWrite(mergedFile);

        if (f1Read.is_open() && f2Read.is_open() && mergedFileWrite.is_open()) {
                mergedFileWrite << f1Read.rdbuf() << f2Read.rdbuf();
                f1Read.close();
                f2Read.close();
                mergedFileWrite.close();
                cout << "The contents of merged file are: ";
                // Reading merged file and displaying content
                ifstream mergedFileRead(mergedFile);
                if (mergedFileRead.is_open()) {
                        cout << mergedFileRead.rdbuf()<<endl;
                        mergedFileRead.close();
                        remove("file1.txt");
                        remove("file2.txt");
                        remove(mergedFile.c_str());
                }
                else {
```

```
                }
        }
        else {
                cout << "Unable to merge files" << endl;
                return 1;
        }
        return 0;
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the data to be inserted in file 1:
Hello world
Enter the data to be inserted in file 2:
How are you?
Merging files...
The contents of merged file are: Hello worldHow are you?

Test Case - 2
User Output
Enter the data to be inserted in file 1:
Hello,
Enter the data to be inserted in file 2:
Friends
Merging files...
The contents of merged file are: Hello,Friends