Experiment No: 9 Date:

Aim: To study file processing

Theory:

1. File Input and Output Streams

C++ provides file input (`ifstream`) and output (`ofstream`) streams to interact with external files. These streams offer methods for reading from and writing to files, facilitating file processing operations.

2. Opening and Closing Files

To work with files, developers use the `open()` method to associate a file with a stream. Proper file closure is achieved through the `close()` method, ensuring data integrity and freeing system resources.

3. Reading from a File

File input streams (`ifstream`) support operations like `>>` and `getline()` to read data from files. These methods enable the extraction of information, such as integers or strings, from the file.

4. Writing to a File

File output streams (`ofstream`) utilize the `<<` operator and `write()` method to write data to files. This allows for the creation and modification of files, as well as the storage of program-generated information.

5. Checking File Status

C++ provides methods like `good()`, `bad()`, `fail()`, and `eof()` to check the status of file streams. These functions help handle different scenarios, such as successful operations or errors during file processing.

6. File Position Pointers

File position pointers (`tellg()` and `seekg()`, or `tellp()` and `seekp()` for output streams) determine and manipulate the current position in a file. This is crucial for navigating and modifying file contents.

7. Binary File I/O

C++ supports reading and writing binary files using `read()` and `write()` functions. This is essential for dealing with non-textual data, ensuring accurate storage and retrieval of binary information.

8. Error Handling in File Processing

Effective error handling is achieved by checking the state of file streams and using exception handling mechanisms. This ensures graceful program behavior in the presence of unexpected file-related issues.

9. File Stream Manipulators

Manipulators like `setw`, `setprecision`, and `std::fixed` enhance control over file output formatting. They allow developers to present data in a structured and readable manner when writing to files.

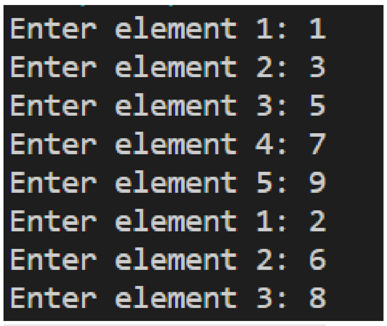
10. File Modes

File modes (`ios::in`, `ios::out`, `ios::app`, etc.) define the intended usage of a file stream (reading, writing, appending). Developers specify these modes during file stream initialization, ensuring proper file access.

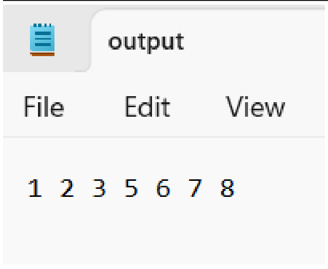
[A] Write a C++ program to insert 5 elements in first file and 3 elements in second file. Merge the contents of both files into third file into ascending order.

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

using namespace std;

void insertElementsToFile(const string& filename, int
numElements) {
  ofstream file(filename);
  if (!file.is_open()) {
    cerr << "Error opening file " << filename << endl;
    return;
  }

  for (int i = 0; i < numElements; i++) {
    int element;
    cout << "Enter element " << i + 1 << ": ";
    cin >> element;
    file << element << " ";
  }

  file.close();
}

void mergeAndSortFiles(const string& file1, const
string& file2, const string& outputFile) {
  ifstream input1(file1);
  ifstream input2(file2);
  ofstream output(outputFile);

  int element1, element2;
  int temp1, temp2; // Temporary variables to store
elements

  if (input1 >> element1 && input2 >> element2) {
    temp1 = element1;
    temp2 = element2;
  }
  while (!input1.eof() && !input2.eof()) {
    if (temp1 < temp2) {
      output << temp1 << " ";
      if (input1 >> temp1) continue; // Read next
element from input1
    } else {
      output << temp2 << " ";
      if (input2 >> temp2) continue; // Read next
element from input2
    }
  }

  while (input1 >> element1) {
    output << element1 << " ";
  }

  while (input2 >> element2) {
    output << element2 << " ";
  }

  input1.close();
  input2.close();
  output.close();
}

int main() {
  insertElementsToFile("file1.txt", 5);
  insertElementsToFile("file2.txt", 3);

  mergeAndSortFiles("file1.txt", "file2.txt",
"output.txt");

  cout << "Merged and sorted elements written to
output.txt" << endl;

  return 0;
}
```

Output:



```
Enter element 1: 1
Enter element 2: 3
Enter element 3: 5
Enter element 4: 7
Enter element 5: 9
Enter element 1: 2
Enter element 2: 6
Enter element 3: 8
```



```
1 2 3 5 6 7 8
```

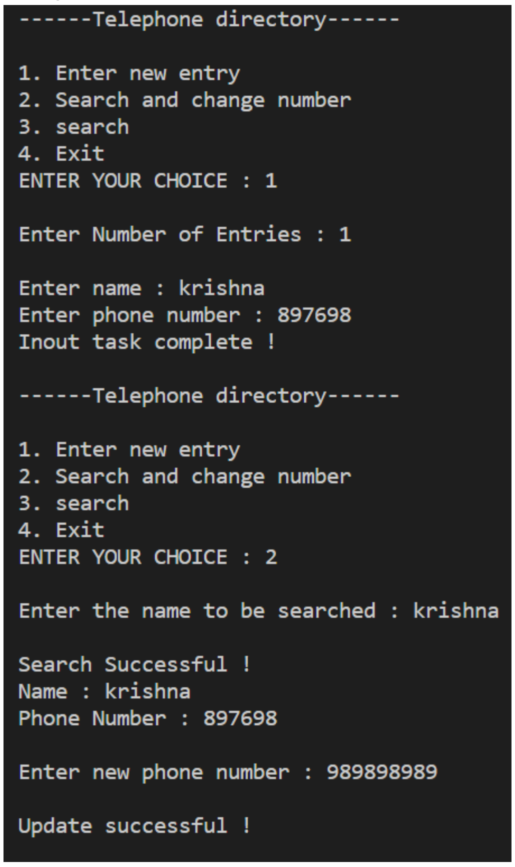
[B] Write a C++ program to simulate a telephone directory application. Program should prompt user to enter name and telephone number of users. Also the program should allow the user to search and update the telephone number of a specific user depending upon the name entered.

```
#include <iostream>
#include <fstream>
#include <string.h>
#include <vector>
```

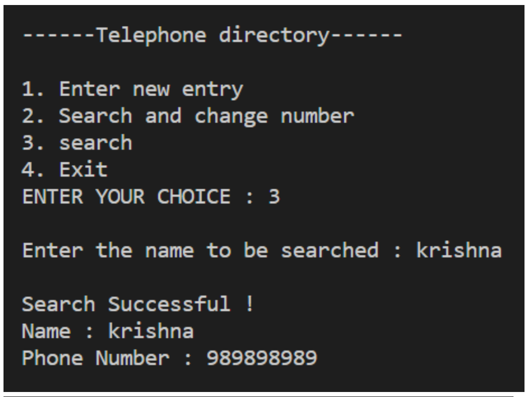
```
#include<stdlib.h>
#include<stdio.h>
#include<cstdlib>

using namespace std;

class TP_directory{
public:
   string name;
   string number;

   static void enterDetails();
   static int searchDetails();
   static void editDetails(int);

};

class IO{
public:
   static void serializeDetails(TP_directory *ptr,
fstream &file);
   static void deserializeDetails(TP_directory *ptr,
fstream &file);
};

void IO :: serializeDetails(TP_directory *ptr,fstream
&file){
   file << ptr->name.size();
   file << ',';
   file << ptr->name.c_str();
   file << ',';
   file << ptr->number.size();
   file << ',';
   file << ptr->number.c_str();
   file << '\n';
}

void IO :: deserializeDetails(TP_directory *ptr,fstream
&file){
   char dlem;
   int len;
   file >> len;
   file >> dlem;
   if (file && len) {
      vector<char> tmp(len);
      file.read(tmp.data() , len);
      ptr->name.assign(tmp.data(), len);
   }
   file >> dlem;
   file >> len;
   file >> dlem;
   if (file && len) {
      vector<char> tmp(len);
      file.read(tmp.data() , len);
      ptr->number.assign(tmp.data(), len);
   }
   //file >> dlem;
}

void TP_directory :: enterDetails(){
   int n;
   TP_directory ent;
   fstream file;
   file.open("TelePhoneBook.txt",ios::in );
   if(!file){
      file.close();
      file.open("TelePhoneBook.txt",ios::out );
   }
   else{
      file.close();
      file.open("TelePhoneBook.txt",ios::app );
   }
   file.seekp(0,ios::end);
   cout<<"\nEnter Number of Entries : ";
   cin >> n;
   while(n--){
      cin.ignore();
      cout<<"\nEnter name : ";
      getline(cin,ent.name);
      cout<<"Enter phone number : ";
      getline(cin,ent.number);
      IO::serializeDetails(&ent,file);
   }
   cout<<"Inout task complete !"<<endl;
}

int TP_directory ::searchDetails(){
   string name;
   int pos = 0, len;
   cout<<"\nEnter the name to be searched : ";
   cin.ignore();
   getline(cin, name);

   TP_directory src;
   fstream file;
   file.open("TelePhoneBook.txt",ios::in );
   if(!file){
      cout<<"\nNo Data is available ! "<<endl;
   }
   file.seekg(0,ios::end);
   len = file.tellg();
   file.seekg(0,ios::beg);

   while(len-=pos){
      pos = file.tellg();
      IO ::deserializeDetails(&src, file);
```

```
        if(src.name == name){
            cout<<"\nSearch Successful ! "<<endl;
            cout<<"Name : "<<src.name<<endl;
            cout<<"Phone Number : "<<src.number<<endl;
            break;
        }
    }
    if(src.name != name){
        cout<<"\nSearch unsuccessful"<<endl;
    }
    file.close();
    return pos;
}

void TP_directory :: editDetails(int edit_pos){
    if(edit_pos < 0){
        return;
    }
    fstream filein,fileout;
    int len, pos = 0;
    TP_directory cp;
    filein.open("TelePhoneBook.txt", ios::in);
    fileout.open("updating.txt", ios::out);
    if(!filein || !fileout){cout<<"Error opening file !
"<<endl;}
    else{
        filein.seekg(0,ios::end);
        len = filein.tellg();
        filein.seekg(0,ios::beg);
        while(!filein.eof()){
            pos = filein.tellg();
            IO ::deserializeDetails(&cp,filein);
            if(filein.fail()) break;
            if(pos == edit_pos){
                cout<<"\nEnter new phone number : ";
                //cin.ignore();
                getline(cin,cp.number);
            }
            IO ::serializeDetails(&cp,fileout);
        }
        filein.close();
        fileout.close();
        char a[] = "TelePhoneBook.txt";
        if(remove(a) != 0)
            cout<<"\nError Deleting old file ! "<<endl;
        if(rename("updating.txt","TelePhoneBook.txt") !=
0)
            cout<<"\nError renaming file !"<<endl;
        cout<<"\nUpdate successful ! "<<endl;
    }
}

int main()
{
    int choice;

    while(1){
        cout<<"\n------Telephone directory------
\n"<<endl;
        cout<<"1. Enter new entry"<<endl;
        cout<<"2. Search and change number"<<endl;
        cout<<"3. search"<<endl;
        cout<<"4. Exit"<<endl;
        cout<<"ENTER YOUR CHOICE : ";
        cin >> choice;
        switch(choice){
        case 1:
            TP_directory::enterDetails();
            break;
        case 2:
            TP_directory
::editDetails(TP_directory::searchDetails());
            break;
        case 3:
            TP_directory::searchDetails();
            break;
        case 4: return 0;
        default: cout<<"wrong input!"<<endl;
        }
    }
}
```

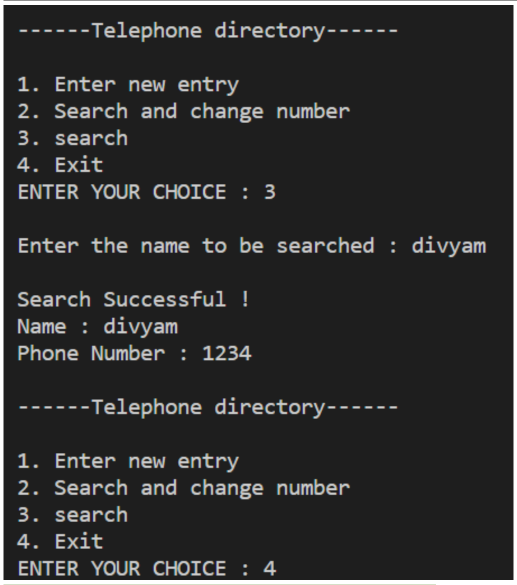
Output:



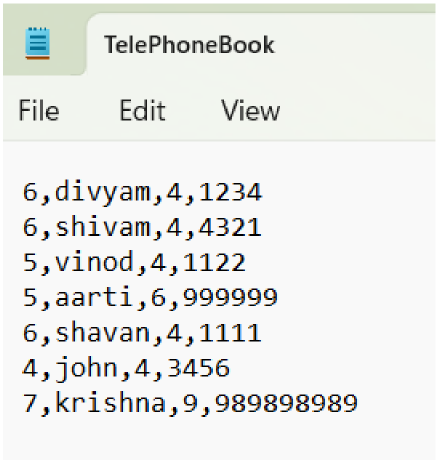
```
------Telephone directory------

1. Enter new entry
2. Search and change number
3. search
4. Exit
ENTER YOUR CHOICE : 1

Enter Number of Entries : 1

Enter name : krishna
Enter phone number : 897698
Inout task complete !

------Telephone directory------

1. Enter new entry
2. Search and change number
3. search
4. Exit
ENTER YOUR CHOICE : 2

Enter the name to be searched : krishna

Search Successful !
Name : krishna
Phone Number : 897698

Enter new phone number : 989898989

Update successful !
```



```
------Telephone directory------

1. Enter new entry
2. Search and change number
3. search
4. Exit
ENTER YOUR CHOICE : 3

Enter the name to be searched : krishna

Search Successful !
Name : krishna
Phone Number : 989898989
```



```
------Telephone directory------

1. Enter new entry
2. Search and change number
3. search
4. Exit
ENTER YOUR CHOICE : 3

Enter the name to be searched : divyam

Search Successful !
Name : divyam
Phone Number : 1234

------Telephone directory------

1. Enter new entry
2. Search and change number
3. search
4. Exit
ENTER YOUR CHOICE : 4
```

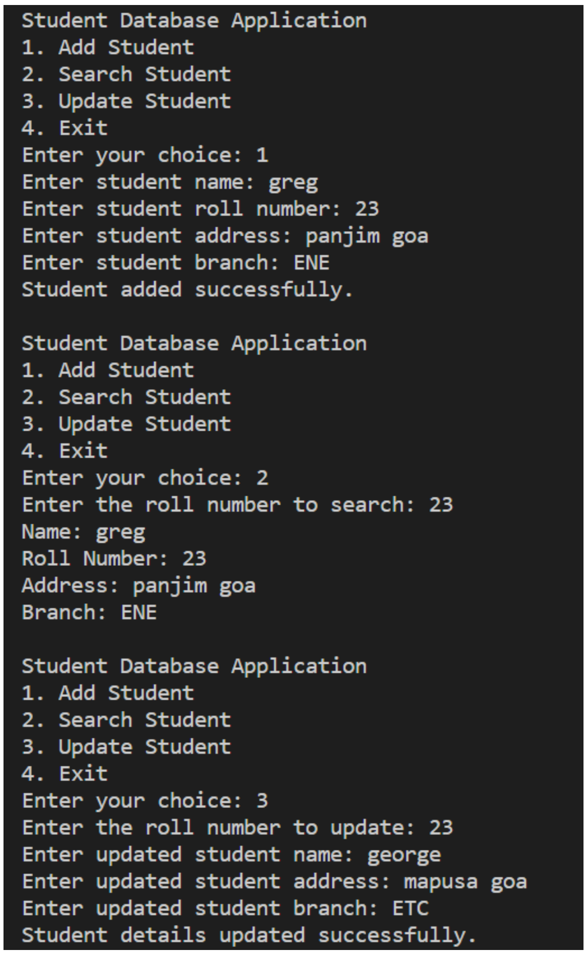


```
6,divyam,4,1234
6,shivam,4,4321
5,vinod,4,1122
5,aarti,6,999999
6,shavan,4,1111
4,john,4,3456
7,krishna,9,989898989
```

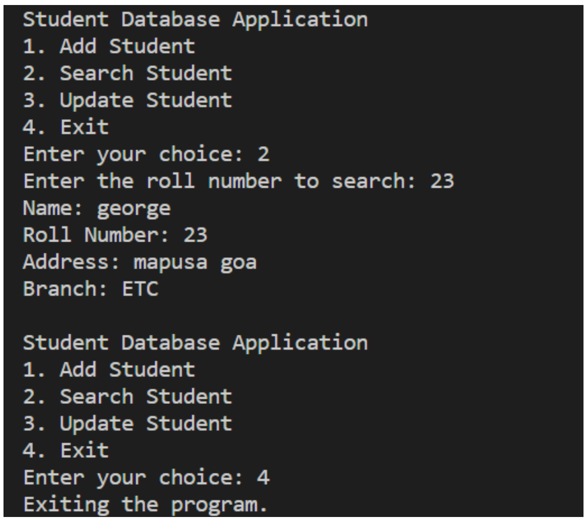
[C] Write a C++ program to create a student's database application using "files". Create a unique file for each student depending upon the student name entered. Store the student data like name, roll no, address, and branch into the file. Allow the user to search and update all the student details depending upon the entered roll-no and display the details.

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

using namespace std;

struct Student {
    string name;
    string rollNo;
    string address;
    string branch;
};

void addStudent() {
    ofstream outFile;
    Student newStudent;

    cout << "Enter student name: ";
    cin >> newStudent.name;

    cout << "Enter student roll number: ";
    cin >> newStudent.rollNo;
    cin.ignore();
    cout << "Enter student address: ";
    getline(cin, newStudent.address);

    cout << "Enter student branch: ";
    cin >> newStudent.branch;

    outFile.open(newStudent.rollNo + ".txt");

    outFile << "Name: " << newStudent.name << endl;
    outFile << "Roll Number: " << newStudent.rollNo << 
endl;
    outFile << "Address: " << newStudent.address << 
endl;
    outFile << "Branch: " << newStudent.branch << 
endl;

    outFile.close();

    cout << "Student added successfully." << endl;
}

void searchStudent() {
    int rollNo;
    cout << "Enter the roll number to search: ";
    cin >> rollNo;

    ifstream inFile;
    string fileName = to_string(rollNo) + ".txt";

    inFile.open(fileName);

    if (inFile.is_open()) {
        string line;
        while (getline(inFile, line) ) {
            cout << line << endl;
        }
        inFile.close();
    } else {
        cout << "Student with roll number " << rollNo << "
not found." << endl;
    }
}

void updateStudent() {
    int rollNo;
    cout << "Enter the roll number to update: ";
    cin >> rollNo;

    fstream file;
    string fileName = to_string(rollNo) + ".txt";

    file.open(fileName, ios::in | ios::out);

    if (file.is_open()) {
        Student updatedStudent;

        cout << "Enter updated student name: ";
        cin >> updatedStudent.name;

        cout << "Enter updated student address: ";
        cin.ignore();
        getline(cin, updatedStudent.address);

        cout << "Enter updated student branch: ";
        cin >> updatedStudent.branch;

        file << "Name: " << updatedStudent.name <<
endl;
        file << "Roll Number: " << rollNo << endl;
        file << "Address: " << updatedStudent.address <<
endl;
        file << "Branch: " << updatedStudent.branch <<
endl;

        file.close();

        cout << "Student details updated successfully." <<
endl;
    } else {
        cout << "Student with roll number " << rollNo << "
not found." << endl;
    }
}

int main() {
    int choice;

    do {
        cout << "\nStudent Database Application\n";
        cout << "1. Add Student\n";
        cout << "2. Search Student\n";
        cout << "3. Update Student\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                addStudent();
                break;
            case 2:
                searchStudent();
                break;
            case 3:
                updateStudent();
                break;
            case 4:
                cout << "Exiting the program.\n";
                break;
            default:
                cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != 4);

    return 0;
}
```

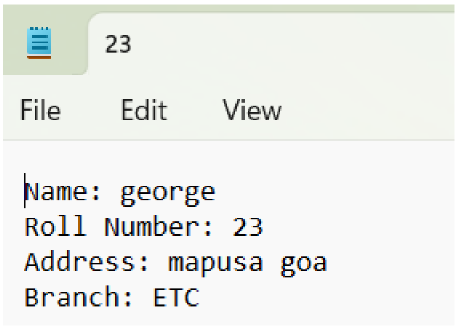
Output:



```
Student Database Application
1. Add Student
2. Search Student
3. Update Student
4. Exit
Enter your choice: 1
Enter student name: greg
Enter student roll number: 23
Enter student address: panjim goa
Enter student branch: ENE
Student added successfully.

Student Database Application
1. Add Student
2. Search Student
3. Update Student
4. Exit
Enter your choice: 2
Enter the roll number to search: 23
Name: greg
Roll Number: 23
Address: panjim goa
Branch: ENE

Student Database Application
1. Add Student
2. Search Student
3. Update Student
4. Exit
Enter your choice: 3
Enter the roll number to update: 23
Enter updated student name: george
Enter updated student address: mapusa goa
Enter updated student branch: ETC
Student details updated successfully.
```



```
Student Database Application
1. Add Student
2. Search Student
3. Update Student
4. Exit
Enter your choice: 2
Enter the roll number to search: 23
Name: george
Roll Number: 23
Address: mapusa goa
Branch: ETC

Student Database Application
1. Add Student
2. Search Student
3. Update Student
4. Exit
Enter your choice: 4
Exiting the program.
```



```
Name: george
Roll Number: 23
Address: mapusa goa
Branch: ETC
```

Conclusion: The concepts and techniques used in file processing in C++ were understood and implemented in the above code snippets.

Nitesh Naik

(Subject Faculty)