Q1. Write a C++ program to create a file and print “File created successfully” and throw an error if file is not created.

```
#include <iostream>
#include <fstream>
#include <conio.h>
using namespace std;
int main() {
	ofstream fout;
	fout.open("D:/abc.txt");
	if(!fout){
		cout<<"File not created";
	}
	else{
		cout<<"File created successfully";
	}
	fout.close();
}
```

Q2. Write a C++ program to read a text file and count the number of characters in it.

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

int main() {

	string text;
	int count = 0;

	ifstream fin;

	fin.open("D:/abc.txt");

	while(getline(fin, text)){
		count = count + text.length();
	}

	cout<<"Length : "<<count;

	fin.close();
}
```

Q3. Write a C++ program to open an output file 'a.txt' and append data to it.

```
#include <iostream>
#include <conio.h>
#include <fstream>
using namespace std;

int main() {
   string txt;
        fstream file;

        file.open("D:/a.txt",ios::app);

        if(!file)
                cout<<"File not open";
        else
                cout<<"File successfuly open"<<endl<<endl;
        do{
                cout<<"Enter data to file : ";
                getline(cin, txt);

                if(txt == "-1")
                        break;
                else
                        file<<txt<<endl;
        }while(file);
        file.close();
}
```

Q4. Write a program to copy the contents of one text file to another while changing the case of every alphabet.

```
#include <iostream>
#include <string>
#include <fstream>
#include <cctype>
using namespace std;

int main() {

    ifstream fin;
    ofstream fout;

    string text;
    char c;

    fin.open("D:/abc.txt", ios::in);
    fout.open("D:/new.txt", ios::app);

    while(getline(fin, text)) {

        for(int i=0; i<text.length(); i++) {
            if(text[i]>=65 && text[i]<=90) {
                c = tolower(text[i]);
                fout<<c;
            } else if(text[i]>=97 && text[i]<=122) {
                c = toupper(text[i]);
                fout<<c;
            }
```

```
        }
    }

    fin.close();
    fout.close();
}
```

Q5. Write a C++ program to merge the two files.

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

int main() {

    ifstream fin;
    ofstream fout;

    string txt;

    fin.open("D:/first.txt", ios::in);
    fout.open("D:/new.txt", ios::app);

    while(getline(fin, txt)){
        fout<<txt<<endl;
    }
    fin.close();
    fin.open("D:/second.txt", ios::in);
    while(getline(fin, txt)){
        fout<<txt<<endl;
    }
    fin.close();
    fout.close();
}
```

Q6. Write a C++ program that counts the total number of characters, words and lines in the file.

```
#include <iostream>

#include<fstream>

#include <string>

using namespace std;

int main() {

   ifstream fin;

   string txt;
   int lines, words, characters, space;

   space=lines=words=characters=0;

   fin.open("D:/abc.txt", ios::in);

   while(getline(fin, txt)) {
      lines = lines + 1;

      for(int i=0; txt[i] != '\0'; i++) {

         if(txt[i] != ' ') {
            characters = characters + 1;
            if(txt[i+1] == '\0'){
               words = words + 1;
            }
         } else {
            words = words + 1;
         }
      }
   }
   cout<<endl<<"Lines = "<<lines<<endl<<"Characters = "<<characters<<endl<<"Words = "<<words;
   fin.close();
}
```

Q7. There are 50 records in a file. Each record contains 6-character item-code, 20 characters for item-name and an integer price. Write a program to read these records, arrange them in the descending order of price and write them in the same file, overwriting the earlier records.

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

class Record{

private:
   string code, name;
   int price;

public:
   void setCode(string C){code = C;}
   void setName(string N){name = N;}
   void setPrice(int P){price = P;}

   string retCode(void){return code;}
   string retName(void){return name;}
   int retPrice(void){return price;}
};

int main() {
   int n = 4;
   Record r[n];
   Record tmp;

   fstream fin;

   fin.open("D:/abc.txt", ios::in);

   string txt;

   for(int i = 0; i < n; i++){
      getline(fin, txt);
      r[i].setCode(txt);

      getline(fin, txt);
      r[i].setName(txt);

      getline(fin, txt);
      r[i].setPrice(stoi(txt));

      getline(fin, txt);
   }

   for(int i = 0; i < n; i++){
      for(int j = i+1; j < n; j++){
         if(r[i].retPrice() < r[j].retPrice()){

            tmp.setCode(r[j].retCode());
            tmp.setName(r[j].retName());
            tmp.setPrice(r[j].retPrice());
```

```
            r[j].setCode(r[i].retCode());
            r[j].setName(r[i].retName());
            r[j].setPrice(r[i].retPrice());

            r[i].setCode(tmp.retCode());
            r[i].setName(tmp.retName());
            r[i].setPrice(tmp.retPrice());
        }
      }
   }
   fin.close();

   ofstream fout;
   fout.open("D:/abc.txt", ios::out);

   for(int i = 0; i < n; i++){
      fout<<r[i].retCode()<<endl;
      fout<<r[i].retName()<<endl;
      fout<<r[i].retPrice()<<endl<<endl;
   }

   fout.close();
}
```

Q8. A file 'Employee.txt' contains empno and empname. Write a C++ program to add and read contents of this file and search for an employee whose name is 'XYZ'.

```
#include <iostream>

#include <fstream>

using namespace std;


int main() {

    int ch=5;


    while(ch!=4) {

        cout<<"1. Add data"<<endl;

        cout<<"2. Read data"<<endl;

        cout<<"3. Search data"<<endl;

        cout<<"4. Exit"<<endl<<endl;

        cout<<"Enter choice = ";

        cin>>ch;


        switch(ch) {


        case 1: {

            ofstream fout;

            fout.open("D:/Employee.txt", ios::app);


            string empno, empname;


            cout<<"Enter emp no = ";

            cin>>empno;


            cout<<"Enter emp name = ";

            cin>>empname;
```

```
    fout<<"emp no = "<<empno<<endl<<"emp name = "<<empname<<endl;

    fout.close();

    break;
}
case 2: {
    ifstream fin;
    fin.open("D:/Employee.txt", ios::in);

    string txt;

    while(getline(fin, txt)) {
        cout<<txt<<endl;
    }

    fin.close();
    break;
}
case 3: {
    ifstream fin;
    fin.open("D:/Employee.txt", ios::in);

    string txt, input;
    cout<<"Enter name = ";
    cin>>input;

    while(getline(fin, txt)) {
        if(txt.find(input) != string::npos) {
            cout<<"emp name found";
```

```
                break;
            }
        }


        fin.close();
        break;
    }
    }
  }
}
```

Q9. A company has following details of their employees in the file 'emp.dat'
a. Emp Id
b. Emp Name
c. Emp Address
d. Emp Dept (Admin/Sales/Production/IT)
e. Emp Phone
f. Emp Age
Write a C++ program to read the above file. Create a new file such as Adm.dat, Sal.dat, Pro.dat, IT.dat respectively to store the employee details according to their department.

```
#include <iostream>
#include <fstream>
#include <string.h>
#include <cstdlib>
#include <conio.h>
#include <cstdio>
#include <limits>

using namespace std;

int options(int);
void printSpecific(void);

class Employee
{
  private:
     string id, name, add, dept, phone, age, space;

  public:
     string const EMPpath = "emp.dat";
     string const ADMpath = "Adm.dat";
     string const SALpath = "Sal.dat";
     string const PROpath = "Pro.dat";
     string const ITpath = "IT.dat";

     int checkFile(string const filepath)
     {
        ifstream fin;
        ofstream fout;

        fin.open(filepath, ios::in);

        if(fin) // checking, file is created or not. if already created then return 1
        {
           fin.close();
           return 1;
        }
        else
```

```
    {
        fout.open(filepath, ios::out); // if file is not created then create new file.
        fout.close();
        return 0;
    }
}
void updateData(string filepath)//this function is use for empUpdateData. for upadint existing user
{
    cin.clear();
    cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');

    ofstream fout;
    ifstream fin;

    fout.open(filepath, ios::app);
    fin.seekg(0, ios::end);

    cout<<"Emp Id: ";
    getline(cin, id);
    fout<<"Emp Id: "<<id<<endl;

    cout<<"Emp Name: ";
    getline(cin, name);
    fout<<"Emp Name: "<<name<<endl;

    cout<<"Emp Age: ";
    getline(cin, age);
    fout<<"Emp Address: "<<age<<endl;

    cout<<"Emp Department: ";
    getline(cin, dept);
    fout<<"Emp Department: "<<dept<<endl;

    cout<<"Emp Phone: ";
    getline(cin, phone);
    fout<<"Emp Phone: "<<phone<<endl;

    cout<<"Emp Address: ";
    getline(cin, add);
    fout<<"Emp Age: "<<add<<endl;

    fout<<"--------------------------------"<<endl;

    fin.close();
    fout.close();
}

void sortData(string filepath)//function for sorting data
{
```

```
    ifstream fin;
    ofstream fout;

    fout.open(filepath, ios::app); //file open in append mode
    fin.seekg(0, ios::end); // move cursor at the end of file

    fout<<id<<endl;
    fout<<name<<endl;
    fout<<age<<endl;
    fout<<dept<<endl;
    fout<<phone<<endl;
    fout<<add<<endl;
    fout<<"--------------------------------"<<endl;

    fout.close();
    fin.close();
}
void print(string const filepath)//display emp data
{
    string line;
    ifstream fin;

    if(checkFile(filepath))
    {
        fin.open(filepath, ios::in); // file open in read mode

        while(getline(fin, line))
        {
            cout<<line<<endl;
        }
        fin.close();
    }
    else
    {
        cout<<"Emp File Is Empty. Enter Data To This File Before Print Operation."<<endl;
        fin.close();
    }
}
void sortEmpData(string const filepath)//sorting emp data
{
    ifstream fin;

    try
    {
        if(checkFile(EMPpath) == 0) // checking emp file is exist or not. if exist the else block execute. if
not the throw error;
        {
            throw string("Emp File Is Empty. Nothing To Sort");
        }
```

```
else
{
   //creating char array for deleting all these old file
    char admpath[] = "Adm.dat";
    char salpath[] = "Sal.dat";
    char propath[] = "Pro.dat";
    char itpath[] = "IT.dat";

   remove(admpath);
   remove(salpath);
   remove(propath);
   remove(itpath);

   checkFile(ADMpath);
   checkFile(SALpath);
   checkFile(ITpath);
   checkFile(PROpath);

   int i = totalData(filepath)/7; // checking data is written or not in emp file. if written then how many entries are there and else block execute. if not then if block execute.

   if(i == 0)
   {
      cout<<"EMP file is empty. No Data Found.";
   }
   else
   {
      fin.open(filepath, ios::in); // file open in read mode

      for(; i; i--)
      {
         getline(fin, id);
         getline(fin, name);
         getline(fin, age);
         getline(fin, dept);
         getline(fin, phone);
         getline(fin, add);
         getline(fin, space);

         if(dept == "Emp Department: Admin")
         {
            sortData(ADMpath);
            continue;
         }
         else if(dept == "Emp Department: Sales")
         {
            sortData(SALpath);
            continue;
         }
```

```
                else if(dept == "Emp Department: IT")
                {
                   sortData(ITpath);
                   continue;
                }
                else if(dept == "Emp Department: Production")
                {
                   sortData(PROpath);
                   continue;
                }
                else
                {
                   //other department category not allow. only admin, sales, it and production is allow.
                }
             }
             fin.close();
             cout<<endl<<"Data Sort Successfuly"<<endl;
          }
       }
    }
    catch(string& err)
    {
       cout<<err;
    }

}
int totalData(string const filepath)
{
    ifstream fin;
    fin.open(filepath, ios::in); // file open in read mode

    int total_data = 0;
    string line;

    while(!fin.eof())// finds how many data are present in emp file.
    {
       getline(fin, line);
       total_data = total_data + 1;
    }

    fin.close();

    return total_data;
}
void updateEmpData(int ch)
{
    ifstream fin;
    ofstream fout;
    int i = 0;
```

```
        if(ch == 1)// update existing data
        {
           try
           {
              if(checkFile(EMPpath))// checking if file is exist or not. if file is not exits then else block
execute.
              {
                 string s1 = "Emp Id: ";
                 string ID;

                 i = totalData(EMPpath)/7;

                 if(i == 0) // checking file is empty or not
                 {
                    cout<<"Emp File Is Empty."<<endl;
                 }
                 else
                 {
                    cout<<"Enter ID To Update Existing Employee Data >>> ";
                    cin>>ID;

                    ifstream ffin;
                    ffin.open(EMPpath, ios::in);

                    for(; i; i--)// get employee data from file
                    {
                       getline(ffin, id);
                       getline(ffin, name);
                       getline(ffin, age);
                       getline(ffin, dept);
                       getline(ffin, phone);
                       getline(ffin, add);
                       getline(ffin, space);

                       if( (s1 + ID) == id) // condition for matching employee id
                       {
                          updateData("tmp.dat"); // if id match then update existing employee
                       }
                       else
                       {
                          sortData("tmp.dat"); // if id not match then copy data from emp.dat to tmp.dat
                       }
                    }
                    ffin.close();
                 }
              }
              else
              {
```

```
                throw string("Emp File Is Empty");
              }
            }
            catch(string& err)
            {
               cout<<err;
            }
         }
         if( (ch == 1 && i != 0) || ch == 4)
         {
            string line;
            char emppath[] = "emp.dat";
            remove(emppath); // before perform this program or this operation please close all program related file(emp.dat, sal.dat, it.dat, pro.dat, adm.dat)
            // delete emp.dat file because all data copy from emp.dat to tmp.dat

            fin.open("tmp.dat", ios::in);
            fout.open(EMPpath, ios::app); //create new emp.dat file

            while(getline(fin, line))//data copy from tmp.dat to emp.dat
            {
               fout<<line<<endl;
            }
            fin.close();
            fout.close();

            char tmppath[] = "tmp.dat";
            remove(tmppath); //tmp.dat file delete because all data copy from tmp.dat to emp.dat

            //creating char array for deleting all these old file
            char admpath[] = "Adm.dat";
            char salpath[] = "Sal.dat";
            char propath[] = "Pro.dat";
            char itpath[] = "IT.dat";

            remove(admpath);
            remove(salpath);
            remove(propath);
            remove(itpath);

            sortEmpData(EMPpath);//after updating employee data, we have to sort employee data for updation in other files
            cout<<"Data Update Successfully"<<endl;
         }
         if(ch == 2)
         {
            updateData(EMPpath);
            sortEmpData(EMPpath);
            cout<<"New Data Update Successfully"<<endl;
```

```
        }
}
void deleteEmpData(string const filepath)
{
   try
   {
      if(checkFile(EMPpath))// checking if file is exist or not. if file is not exits then else block execute.
      {
         string s1 = "Emp Id: ";
         string ID;

         int i = totalData(EMPpath)/7;

         if(i == 0) // checking file is empty or not
         {
            cout<<"Emp File Is Empty."<<endl;
         }
         else
         {
            cout<<"Enter ID To Delete Employee Data >>> ";
            cin>>ID;

            ifstream ffin;
            ffin.open(EMPpath, ios::in);

            for(; i; i--)// get employee data from file
            {
               getline(ffin, id);
               getline(ffin, name);
               getline(ffin, age);
               getline(ffin, dept);
               getline(ffin, phone);
               getline(ffin, add);
               getline(ffin, space);

               if( (s1 + ID) == id )
               {
                  cout<<id<<endl;
                  cout<<name<<endl;
                  cout<<age<<endl;
                  cout<<dept<<endl;
                  cout<<phone<<endl;
                  cout<<add<<endl;
                  cout<<space<<endl<<endl;

                  cout<<"Data Found....Press Enter To Delete...."<<endl;
                  getch();
               }
               else
```

```
                    {
                        sortData("tmp.dat");
                    }
                }
                ffin.close();
                updateEmpData(4);
            }
        }
        else
        {
            throw string("Emp File Is Empty");
        }
    }
    catch(string& err)
    {
        cout<<err;
    }
  }
};

int main()
{
   Employee e;

   int choice = 0;

   system("cls");

   do{
      cout<<"1. Print Data"<<endl;
      cout<<"2. Sort Data"<<endl;
      cout<<"3. Update Data"<<endl;
      cout<<"4. Delete Data"<<endl;
      cout<<"5. Exit"<<endl<<endl;

      cout<<"Enter Your Choice >>> ";
      cin>>choice;

      system("cls");

   }while(!(options(choice) && choice == 5));
}

int options(int choice)//in this function there is a switch case for program execution. this function is non-member function of Employee class.
{
   switch(choice)
   {
   case 1:
```

```
    {
        int ch;
        Employee e;

        cout<<"1. Print All Employee Data"<<endl;
        cout<<"2. Print Specific Employee Data"<<endl<<endl;

        cout<<"Enter Your Choice >> ";
        cin>>ch;

        if(ch == 1)
        {
            system("cls");
            e.print(e.EMPpath);
        }
        else
        {
            printSpecific();
        }

        getch();
        system("cls");
        break;
    }
case 2:
    {
        Employee e;
        e.sortEmpData(e.EMPpath);

        getch();
        system("cls");
        break;
    }
case 3:
    {
        system("cls");

        Employee e;
        int ch = 0;

        cout<<"1. Update Existing Data"<<endl;
        cout<<"2. Enter New Data"<<endl<<endl;

        cout<<"Enter Choice >>> ";
        cin>>ch;

        if(ch == 1)
        {
            e.updateEmpData(1);
```

```
            getch();
            system("cls");
        }
        else if(ch == 2)
        {
            e.updateEmpData(2);
            getch();
            system("cls");
        }
        else
        {
            cout<<"Wrong Choice";
        }

        getch();
        system("cls");
        break;
      }
   case 4:
      {
        system("cls");
        Employee e;

        e.deleteEmpData(e.EMPpath);

        cout<<"Data Delete Successfully";
        system("cls");
        getch();
        break;
      }
   }
   return 1;
}

void printSpecific()// this function is a part of switch case 1.
{
   Employee e;

   system("cls");
   int ch = 0;

   cout<<"1. Print Admin Department Data"<<endl;
   cout<<"2. Print Sales Department Data"<<endl;
   cout<<"3. Print Production Department Data"<<endl;
   cout<<"4. Print IT Department Data"<<endl<<endl;

   cout<<"Enter Your Choice >>> ";
   cin>>ch;
```

```
    system("cls");

    switch(ch)
    {
    case 1:
       {
          e.print(e.ADMpath);
          break;
       }
    case 2:
       {
          e.print(e.SALpath);
          break;
       }
    case 3:
       {
          e.print(e.PROpath);
          break;
       }
    case 4:
       {
          e.print(e.ITpath);
          break;
       }
    default:
       {
          cout<<"Wrong Choice";
          break;
       }
    }

    getch();
    system("cls");
}
```

Q10. Write a C++ program to create a file which has information Name, Account number, Balance and perform following operations:
a. Add record
b. Display content of file
c. Display name of person having balance > 10,000

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

class Person
{
private:
    string name, acc, bal;
    string person = "person.dat";
    ifstream fin;
    ofstream fout;

public:
    void checkFile()
    {
        fin.open(person, ios::in);

        if(fin)
        {
            fin.close();
        }
        else
        {
            fout.open(person, ios::out);
            fout.close();
            cout<<"File is empty";
        }
    }
    void addRecord()
    {
        checkFile();

        fout.open(person, ios::app);
        fout.seekp(0, ios::end);

        cout<<"Enter Name: ";
        cin>>name;
        fout<<"Name: "<<name<<endl;

        cout<<"Enter Account Number: ";
        cin>>acc;
        fout<<"Account Number: "<<acc<<endl;
```

```
    cout<<"Enter Balance: ";
    cin>>bal;
    fout<<"Balance: "<<bal<<endl;

    fout<<"--------------------------------"<<endl;

    fout.close();
}
void display()
{
    checkFile();

    fin.open(person, ios::in);

    string line;

    while(getline(fin, line))
    {
        cout<<line<<endl;
    }
    fin.close();
}
void checkBalance()
{
    checkFile();

    fin.open(person, ios::in);

    int total_data = 0;

    string line;

    while(getline(fin, line))
    {
        total_data = total_data + 1;
    }

    fin.clear();
    fin.seekg(0, ios::beg);

    for(int i = total_data; i; i--)
    {
        getline(fin, name);
        getline(fin, acc);
        getline(fin, bal);
        getline(fin, line);

        if(stoi(bal.substr(9)) > 10000)
```

```
            {
                cout<<name<<endl;
            }
            else
            {
                continue;
            }
        }

        fin.close();
    }

};

int main()
{
    int ch = 1;
    Person p;

    while(ch != 4)
    {
        cout<<"1. Add record"<<endl;
        cout<<"2. Display content of file"<<endl;
        cout<<"3. Display name of person having balance > 10,000"<<endl;
        cout<<"4. Exit"<<endl<<endl;

        cout<<"Enter choice >>> ";
        cin>>ch;

        switch(ch)
        {
        case 1:
            {
                p.addRecord();
                break;
            }
        case 2:
            {
                p.display();
                break;
            }
        case 3:
            {
                p.checkBalance();
                break;
            }
        case 4:
            {
                exit(0);
```

```
        }
      }
    }
}
```