Dayananda Sagar Academy of Technology & Management

Opp. Art of living, Udayapura, Kanakapura road, Bengaluru- 560082
(Affiliated to VTU, Belagavi & Approved by AICTE, New Delhi)

Department of Information Science & Engineering
Accredited by NBA, New Delhi

2021-2022

File Structures Laboratory with Mini Project

Laboratory Manual

18ISL67

Compiled by:
Prof. Babu Kumar S
Prof. Supriya R K
Prof. Neha Jadhav

TABLE OF CONTENTS

INSTITUTION VISION AND MISSION

Vision of the Institution

To strive at creating the institution a center of highest caliber of learning, so as to create an overall intellectual atmosphere with each deriving strength from the other to be the best of engineers, scientists with management &design skills.

Mission of the Institution:

- To serve its region, state, the nation and globally by preparing students to make
- meaningful contributions in an increasing complex global society challenges.
- To encourage, reflection on and evaluation of emerging needs and priorities with state of art infrastructure at institution.
- To support research and services establishing enhancements in technical, health, economic, human and cultural development.
- To establish inter disciplinary center of excellence, supporting/ promoting student's implementation.
- To increase the number of Doctorate holders to promote research culture on campus.
- To establish IIPC, IPR, EDC, innovation cells with functional MOU's supporting student's quality growth.

QUALITY POLICY

Dayananda Sagar Academy of Technology and Management aims at achieving academic excellence through continuous improvement in all spheres of Technical and Management education. In pursuit of excellence cutting-edge and contemporary skills are imparted to the utmost satisfaction of the students and the concerned stakeholders

OBJECTIVES & GOALS

- Creating an academic environment to nurture and develop competent entrepreneurs, leaders and professionals who are socially sensitive and environmentally conscious.
- Integration of Outcome Based Education and cognitive teaching and learning strategies to enhance learning effectiveness.
- Developing necessary infrastructure to cater to the changing needs of Business and Society.
- Optimum utilization of the infrastructure and resources to achieve excellence in all areas of relevance.
- Adopting learning beyond curriculum through outbound activities and creative assignments.
- Imparting contemporary and emerging techno-managerial skills to keep pace with the changing global trends.
- Facilitating greater Industry-Institute Interaction for skill development and employability enhancement.
- Establishing systems and processes to facilitate research, innovation and entrepreneurship for holistic development of students.
- Implementation of Quality Assurance System in all Institutional processes

DAYANANDA SAGAR ACADEMY OF TECHNOLOGY & MANAGEMENT

Opp. Art of Living, Udayapura, Kanakapura Road, Bangalore- 560082

(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi)

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

Accredited by NBA, New Delhi

VISION OF THE DEPARTMENT

Impart magnificent learning atmosphere establishing innovative practices among the students aiming to strengthen their software application knowledge and technical skills.

.

MISSION OF THE DEPARTMENT

M1: To deliver quality technical training on software application domain.

M2: To nurture teamwork in order to transform individual as responsible leader and entrepreneur for future trends.

M3: To inculcate research practices in teaching thus ensuring research blend among students.

M4: To ensure more doctorates in the department, aiming at professional strength.

M5: To inculcate the core information science engineering practices with hardware blend by providing advanced laboratories.

M6: To establish innovative labs, start-ups and patent culture.

.

Program Educational Objectives (PEOs)

PEO1: Graduates shall have successful careers as information science engineers and will be able to lead and manage teams across the globe.

PEO2: Graduates shall be professional in engineering practice and shall demonstrate good problem solving, communication skills and contribute to address societal issues.

PEO3: Graduates shall be pursuing distinctive education, entrepreneurship and research in an excellent environment which helps in the process of life-long learning.

Program Outcomes (POs)

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12. Life-long learning: Recognize the need for, and have the preparation and ability to engage independent and life-long learning in the broadest context of technological change.

\

DAYANANDA SAGAR ACADEMY OF TECHNOLOGY & MANAGEMENT
(Affiliated to Visvesvaraya Technological University,Belagavi & Approved by AICTE,New Delhi)
Opp. Art of Living, Udayapura, Kanakapura Road, Bangalore- 560082
DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING
Accredited by NBA, New Delhi

SUBJECT: File Structures Laboratory with Mini Project

SUBJECT CODE: 18ISL67

SEMESTER: VI

Course Outcomes

At the end of the course the student will be able to:

CO1	Become effective and independent programmers to implement file operations.
CO2	Apply the concepts of file structure organization to produce the given application.
CO3	Evaluate performance of various file structures on given parameters.
CO4	Implementation of a file structure application.

FILE STRUCTURES LABORATORY WITH MINI PROJECT
[As per Choice Based Credit System (CBCS) scheme]
(Effective from the academic year 2018 -2019)
SEMESTER – VI

Subject Code	**18ISL67**	**Internal**	**40 Marks**
Number of Lecture Hours/Week	**01 I + 02 P**	**External**	**60 Marks**
Total Lecture Hours	**40**	**Duration of Exam**	**3 Hours**

CREDITS – 02

Course objectives: This course will enable students to

- Become effective and independent programmers to implement file operations.
- Apply the concepts of file structure organization to produce the given application.
- Evaluate performance of various file structures on given parameters.
- Implementation of a file structure application.

Description (If any):

Design, develop, and implement the following programs

Lab Experiments:

1. Write a program to read series of names, one per line, from standard input and write these names spelled in reverse order to the standard output using I/O redirection and pipes. Repeat the exercise using an input file specified by the user instead of the standard input and using an output file specified by the user instead of the standard output.
2. Write a program to read and write student objects with fixed-length records and the fields delimited by "|". Implement pack ( ), unpack ( ), modify ( ) and search ( ) methods.
3. Write a program to read and write student objects with Variable - Length records using any suitable record structure. Implement pack ( ), unpack ( ), modify ( ) and search ( ) methods.
4. Write a program to write student objects with Variable - Length records using any suitable record structure and to read from this file a student record using RRN.
5. Write a program to implement simple index on primary key for a file of student objects. Implement add ( ), search ( ), delete ( ) using the index.
6. Write a program to implement index on secondary key, the name, for a file of student objects. Implement add ( ), search ( ), delete ( ) using the secondary index.
7. Write a program to read two lists of names and then match the names in the two lists using Consequential Match based on a single loop. Output the names common to both the lists.
8. Write a program to read k Lists of names and merge them using k-way merge algorithm with k =8.

Part B --- Mini project:

Student should develop mini project on the topics mentioned below or similar applications Document processing, transaction management, indexing and hashing, buffer management, configuration management. Not limited to these

Course outcomes: The students should be able to:

- Implement operations related to files
- Apply the concepts of file system to produce the given application.
- Evaluate performance of various file systems on given parameters.

Conduction of Practical Examination:

1. All laboratory experiments from part A are to be included for practical examination.
2. Report should be prepared in a standard format prescribed for project work.
3. Students are allowed to pick one experiment from the lot.
4. Strictly follow the instructions as printed on the cover page of answer script.
5. Marks distribution:

a) Part A: Procedure + Conduction + Viva:04+ 21 +05 =30 Marks

b) Part B: Demonstration + Report + Viva voce = 10+49+11 = 70 Marks

6. Change of experiment is allowed only once and marks allotted to the procedure part to be made zero.

1. Write a C++ program to read series of names, one per line, from standard input and write these names spelled in reverse order to the standard output using I/O redirection and pipes. Repeat the exercise using an input file specified by the user instead of the standard input and using an output file specified by the user instead of the standard output.

I/O redirection: There are always three default files open, stdin (the keyboard), stdout (the screen), and stderr (error messages output to the screen). These, and any other open files, can be redirected. Redirection simply means capturing output from a file, command, program, script, or even code block within a script and sending it as input to another file, command, program or script.

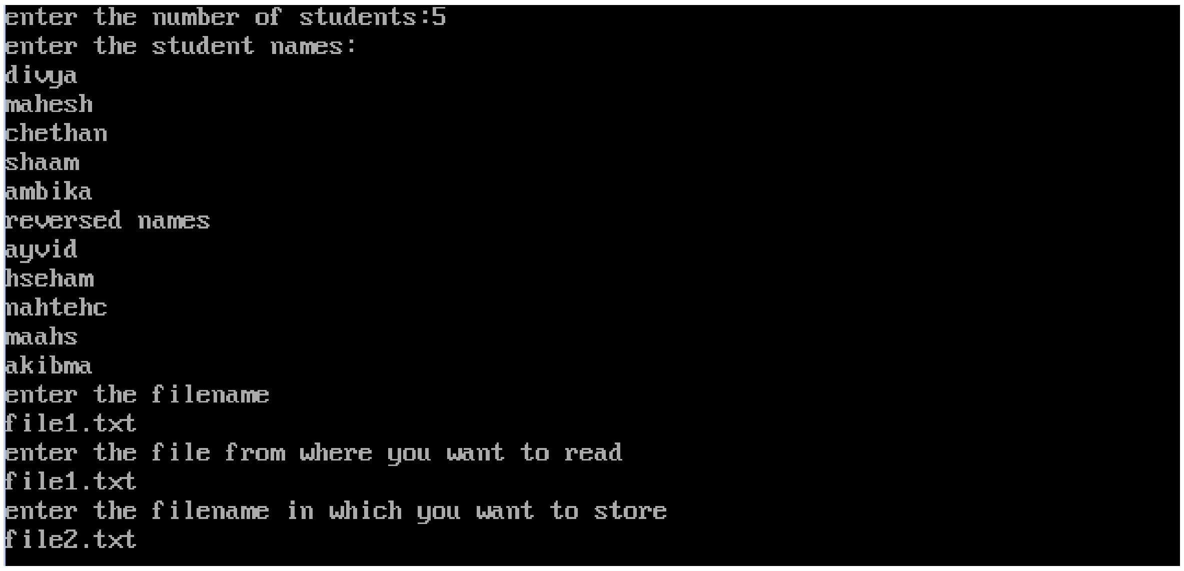
Pipes:A pipe is nothing but a temporary storage place where the output of one command is stored and then passed as the input for second command. Pipes are used to run more than two commands (multiple commands) from same command line.

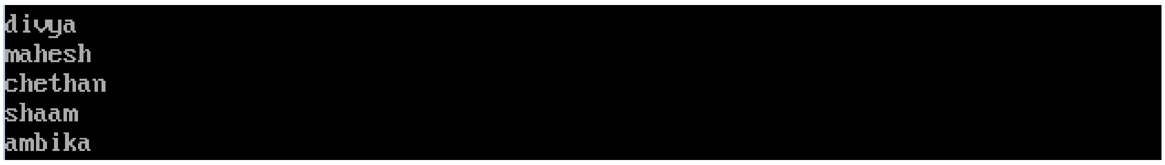
```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
#include<string.h>
#include<stdlib.h>

class student
{
        public: char name[10];
} rec1[10],rec2[10];

int n;

void reads()
{
        char name[10];
        cout<<"enter the number of students:";
        cin>>n;
        cout<<"enter the student names:\n";
        for(int i=0;i<n;i++)
        {
                cin>>rec1[i].name;
        }
        cout<<"reversed names\n";
        for(i=0;i<n;i++)
        {
                strcpy(name,rec1[i].name);
                strrev(name);
                strcpy(rec2[i].name,name);
                cout<<rec2[i].name<<"\n";
        }
}
```

```
void write()
{
        fstream file;
        char fname[10];
        cout<<"enter the filename\n";
        cin>>fname;
        file.open(fname,ios::out);
        if(!file)
        {
                cout<<"could not open the file\n";
                exit(1);
        }
        for(int i=0;i<n;i++)
        {
                file<<rec2[i].name<<"\n";

}
}
```

```
void stored_names()
{
	fstream f1,f2;
	char fname1[10],fname2[10],name[10];
	cout<<"enter the file from where you want to read\n";
	cin>>fname1;
	f1.open(fname1,ios::in);
	if(!f1)
	{
		cout<<"could not open the file\n";
		exit(1);
	}
	cout<<"enter the filename in which you want to store\n";
	cin>>fname2;
	f2.open(fname2,ios::out);
	while(!f1.eof())
	{
		f1.getline(name,10,'\n');
		strrev(name);
		cout<<name<<"\n";
		f2<<name<<"\n";
	}
	f1.close();
	f2.close();
}

void main()
{
	clrscr();
	reads();
	write();
	stored_names();
	getch();
}
```

OUTPUT:


```
enter the number of students:5
enter the student names:
divya
mahesh
chethan
shaam
ambika
reversed names
ayvid
hseham
nahtehc
maahs
akibma
enter the filename
file1.txt
enter the file from where you want to read
file1.txt
enter the filename in which you want to store
file2.txt
```


```
divya
mahesh
chethan
shaam
ambika
```

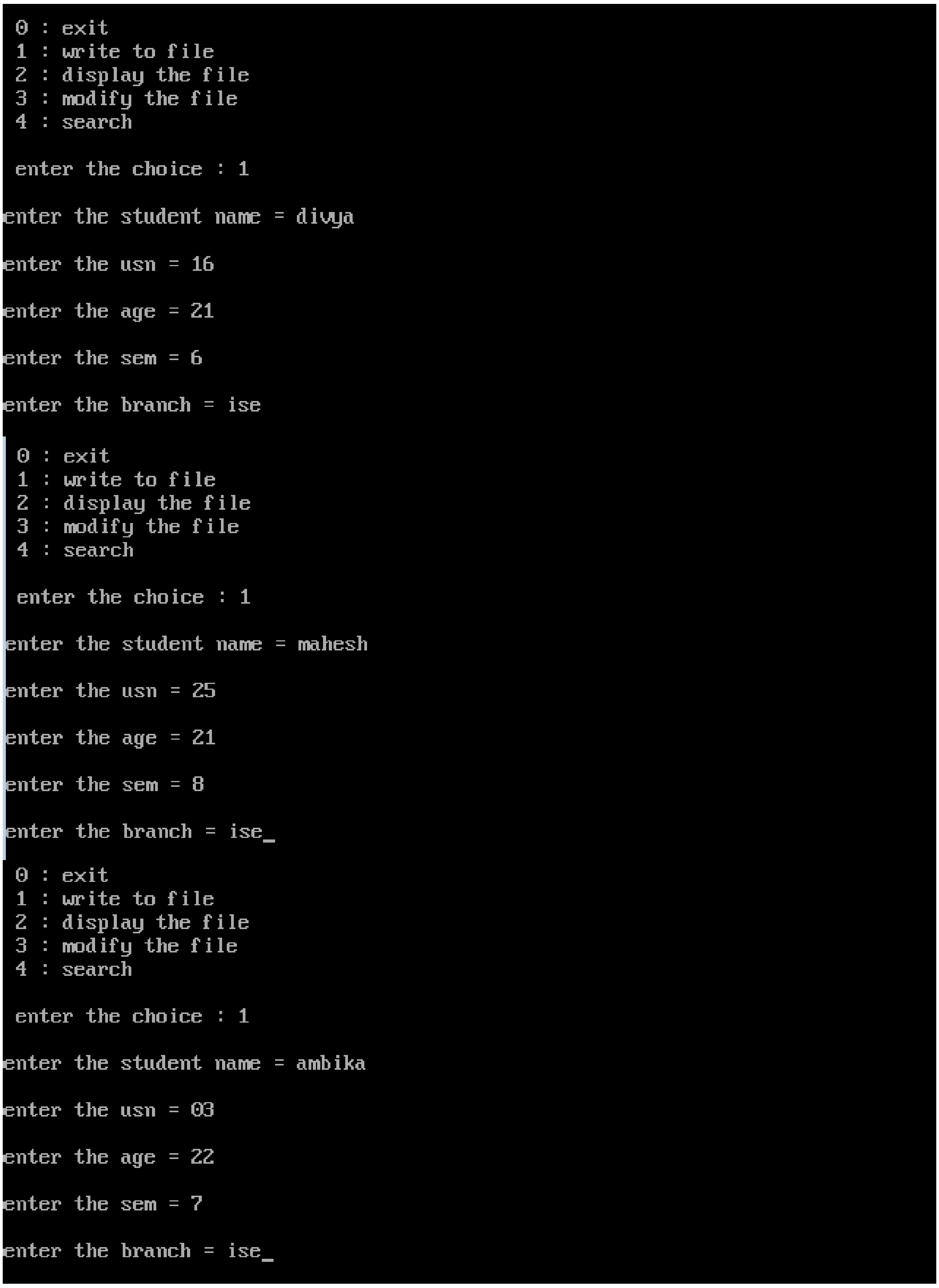
2. Write a C++ program to read and write student objects with fixed-length records and the fields delimited by “|”. Implement pack ( ), unpack ( ), modify ( ) and search ( ) methods.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<iostream.h>
#include<fstream.h>
#include<string.h>

class student
{
        public: char name[10];
                char usn[10];
                char age[5];
                char sem[5];
                char branch[5];
                char buffer[45];
};

fstream file;
student s;
void writerecord()
{
        file.open("student.txt",ios::app);
        if(!file)
        {
                cout<<"cannot open the file in append mode";
                exit(1);
        }
        cout<<"\nenter the student name = ";
        cin>>s.name;
        cout<<"\nenter the usn = ";
        cin>>s.usn;
        cout<<"\nenter the age = ";
        cin>>s.age;
        cout<<"\nenter the sem = ";
        cin>>s.sem;
        cout<<"\nenter the branch = ";
        cin>>s.branch;
        //packing the information
        strcpy(s.buffer,s.name);
        strcat(s.buffer,"|");
        strcat(s.buffer,s.usn);
        strcat(s.buffer,"|");
        strcat(s.buffer,s.age);
        strcat(s.buffer,"|");
        strcat(s.buffer,s.sem);
        strcat(s.buffer,"|");
        strcat(s.buffer,s.branch);

        int count=strlen(s.buffer);
        for(int k=0;k<45-count;k++)
```

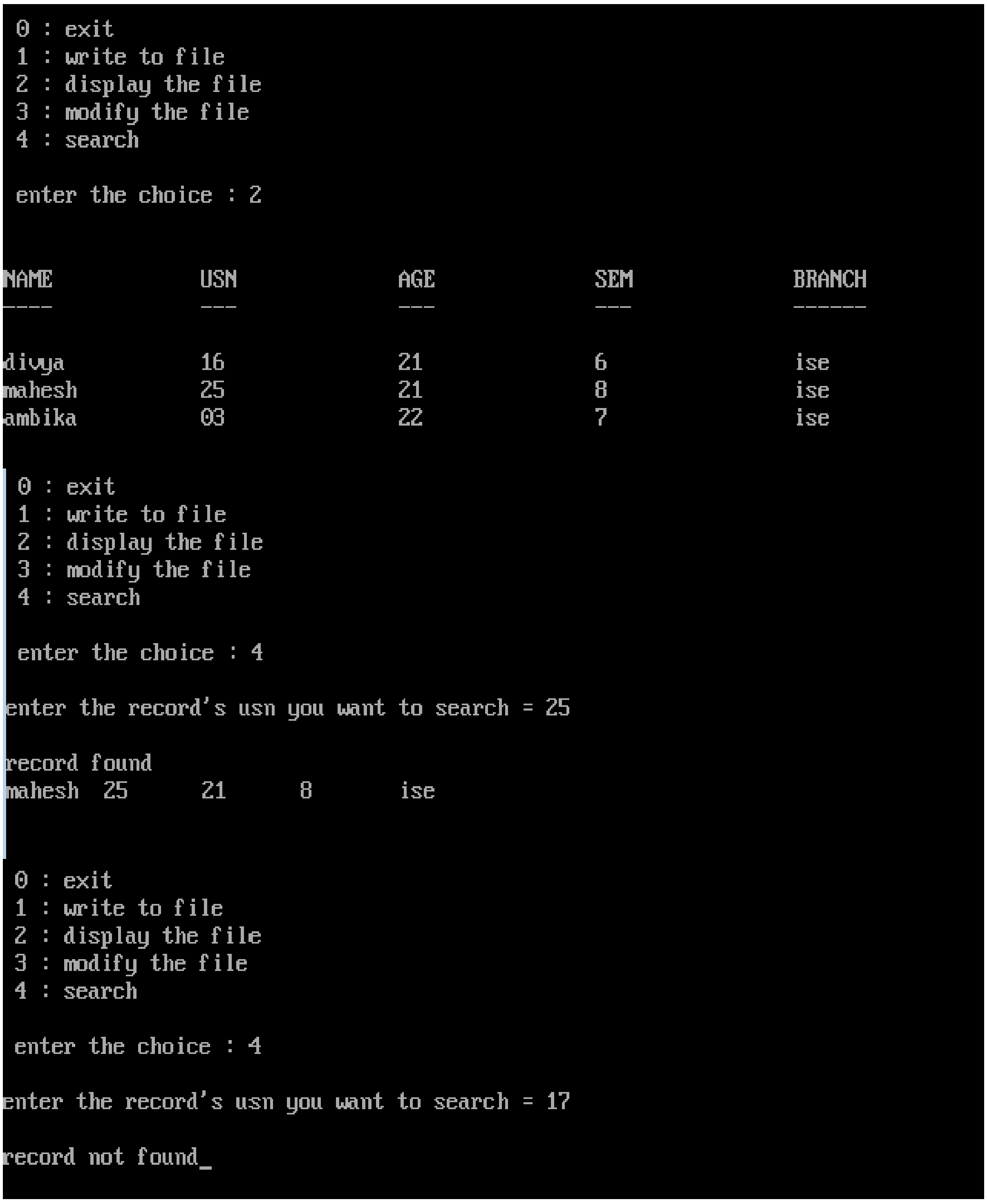
```
			strcat(s.buffer,"!");
		strcat(s.buffer,"\n");
		file<<s.buffer; //writing the packed information to buffer
		file.close();
}

void search()
{
		char usn[10];
		char extra[45];
		file.open("student.txt",ios::in);
		if(!file)
		{
			cout<<"\nunable to open the file in read mode";
			exit(0);
		}
		cout<<"\nenter the record's usn you want to search = ";
		cin>>usn;

		//unpacking the record

		while(!file.eof())
		{
			file.getline(s.name,10,'|');
			file.getline(s.usn,10,'|');
			file.getline(s.age,5,'|');
			file.getline(s.sem,5,'|');
			file.getline(s.branch,5,'!');
			file.getline(extra,45,'\n');

			if(strcmp(s.usn,usn)==0)
			{
				cout<<"\nrecord found";
				cout<<"\n"<<s.name<<"\t"<<s.usn<<"\t";
				cout<<s.age<<"\t"<<s.sem<<"\t"<<s.branch;
				file.close();
				getch();
				return;
			}
		}
		cout<<"\nrecord not found";
		file.close();
		getch();
}

void displayFile()
{
		int i;
		char extra[45];
		file.open("student.txt",ios::in);

		if(!file)
		{
			cout<<"\ncannot open the file in read mode";
```

```
                getch();
                exit(1);
        }
        i=0;
        cout<<"\n\nNAME\t\tUSN\t\tAGE\t\tSEM\t\tBRANCH\n";
        cout<<"----\t\t---\t\t---\t\t---\t\t -----\n";

        while(!file.eof())
        {
                file.getline(s.name,10,'|');
                file.getline(s.usn,10,'|');
                file.getline(s.age,5,'|');
                file.getline(s.sem,5,'|');
                file.getline(s.branch,5,'!');
                file.getline(extra,45,'\n');
                printf("\n%s\t\t%s\t\t%s\t\t%s\t\t%s",s.name,s.usn,s.age,s.sem,s.branch);
                i++;
        }
        file.close();
        getch();
}

void modify()
{
        char usn[10];
        char buffer[45];
        char extra[45];
        int i;
        int j;
        student s[20];

        file.open("student.txt",ios::in);
        if(!file)
        {
                cout<<"\nunable to open the file in input mode";
                getch();
                exit(1);
        }
        cout<<"\nenter the usn of the record to be modified\n";
        cin>>usn;
        cout<<"\n";

        i=0;
        while(!file.eof())
        {
                file.getline(s[i].name,10,'|');
                file.getline(s[i].usn,10,'|');
                file.getline(s[i].age,5,'|');
                file.getline(s[i].sem,5,'|');
                file.getline(s[i].branch,5,'!');
                file.getline(extra,45,'\n');
                i++;
        }
        i--;
        for(j=0;j<i;j++)
        {
                if(strcmp(usn,s[j].usn)==0)
                {
                        cout<<"\nthe old values of the record with usn"<<usn<<"are";
                        cout<<"\nname = "<<s[j].name;
                        cout<<"\nusn = "<<s[j].usn;
                        cout<<"\nage = "<<s[j].age;
                        cout<<"\nsem = "<<s[j].sem;
                        cout<<"\nbranch = "<<s[j].branch;

                        cout<<"\n\nenter the new values\n";
                        cout<<"\nname = ";
                        cin>>s[j].name;
                        cout<<"\nusn = ";
                        cin>>s[j].usn;
                        cout<<"\nage = ";
                        cin>>s[j].age;
                        cout<<"\nsem = ";
                        cin>>s[j].sem;
                        cout<<"\nbranch = ";
                        cin>>s[j].branch;
                        break;
                }
        }

        if(j==i)
        {
                cout<<"\nthe record with usn " <<usn<< "is not present ";
                getch();
                return;
        }
        file.close();

        file.open("student.txt",ios::out);
        if(!file)
        {
                cout<<"\nunable to open the file in output mode";
                getch();
                return;
        }

        for(j=0;j<i;j++)
        {
                strcpy(buffer,s[j].name);
                strcat(buffer,"|");
                strcat(buffer,s[j].usn);
                strcat(buffer,"|");
                strcat(buffer,s[j].age);
                strcat(buffer,"|");
                strcat(buffer,s[j].sem);
                strcat(buffer,"|");
                strcat(buffer,s[j].branch);
                int count=strlen(buffer);
                for(int k=0;k<45-count;k++)
                        strcat(buffer,"!");
                strcat(buffer,"\n");
                file<<buffer;
        }
        file.close();
}

void main()
{
        int choice;
        while(1)
        {
                clrscr();
                cout<<"\n 0 : exit";
                cout<<"\n 1 : write to file";
                cout<<"\n 2 : display the file";
                cout<<"\n 3 : modify the file";
                cout<<"\n 4 : search";
                cout<<"\n\n enter the choice : ";
                cin>>choice;

                switch(choice)
                {
                        case 1: writerecord();
                                break;

                        case 2: displayFile();
                                break;

                        case 3: modify();
                                break;

                        case 4: search();
                                break;

                        case 0: exit(0);

                        default:cout<<"\ninvalid input...";
                                break;
                }
        }
}
```

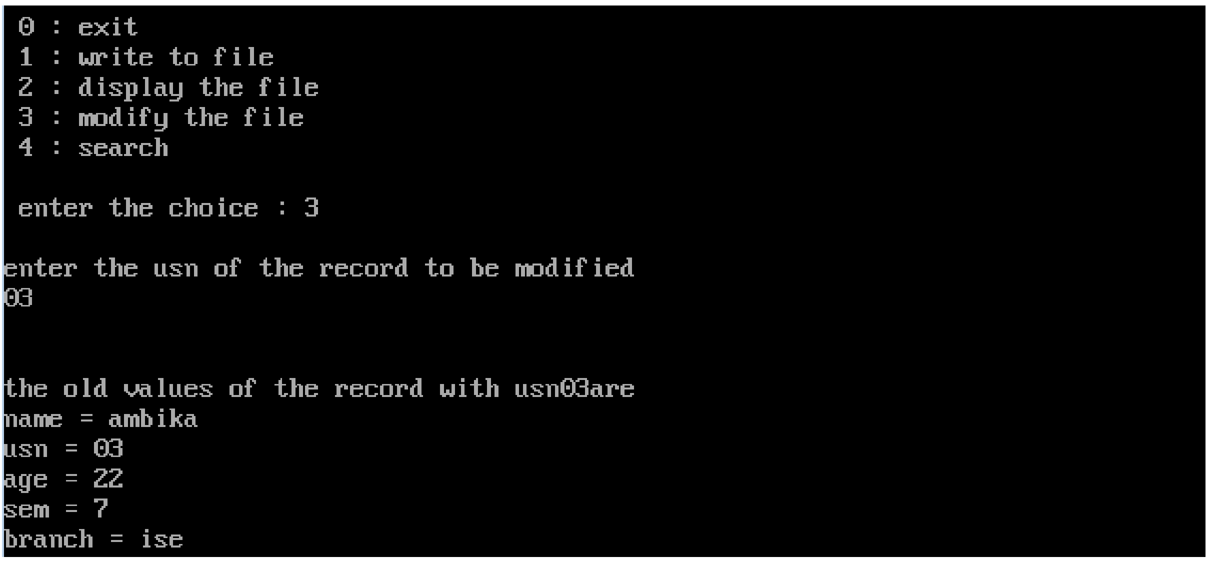
OUTPUT:



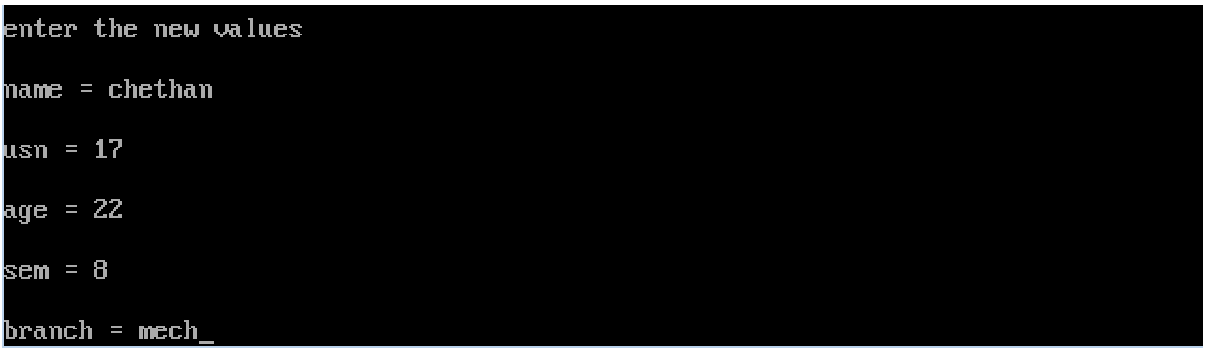
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 1

enter the student name = divya

enter the usn = 16

enter the age = 21

enter the sem = 6

enter the branch = ise

 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 1

enter the student name = mahesh

enter the usn = 25

enter the age = 21

enter the sem = 8

enter the branch = ise_

 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 1

enter the student name = ambika

enter the usn = 03

enter the age = 22

enter the sem = 7

enter the branch = ise_
```



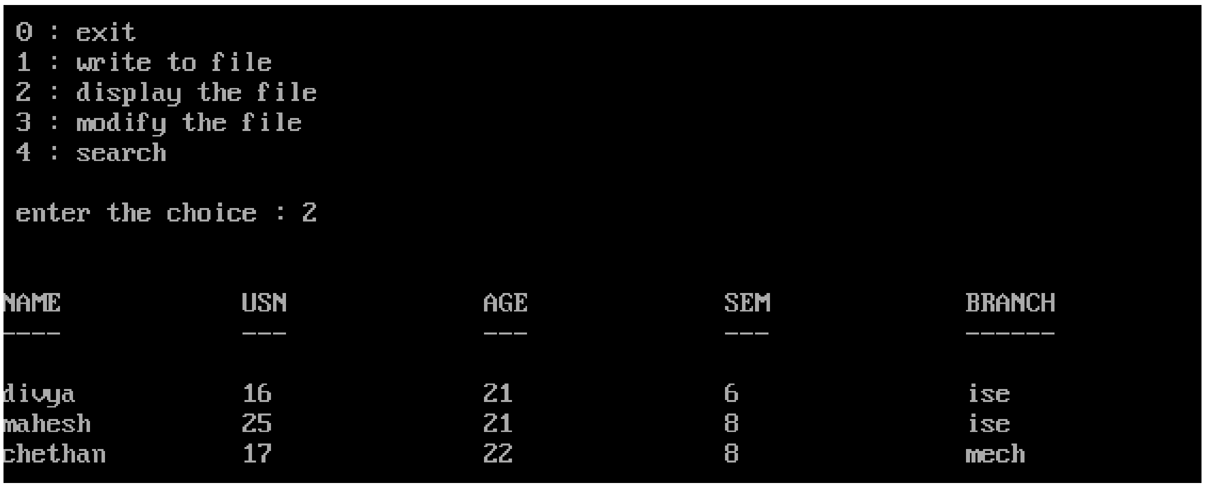
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 2


NAME           USN            AGE            SEM            BRANCH
----           ---            ---            ---            ------

divya          16             21             6              ise
mahesh         25             21             8              ise
ambika         03             22             7              ise

 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 4

enter the record's usn you want to search = 25

record found
mahesh  25      21      8       ise


 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 4

enter the record's usn you want to search = 17

record not found_
```



```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 3

enter the usn of the record to be modified
03


the old values of the record with usn03are
name = ambika
usn = 03
age = 22
sem = 7
branch = ise
```



```
enter the new values

name = chethan

usn = 17

age = 22

sem = 8

branch = mech_
```



```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 2


NAME            USN             AGE             SEM             BRANCH
----            ---             ---             ---             ------

divya           16              21              6               ise
mahesh          25              21              8               ise
chethan         17              22              8               mech
```

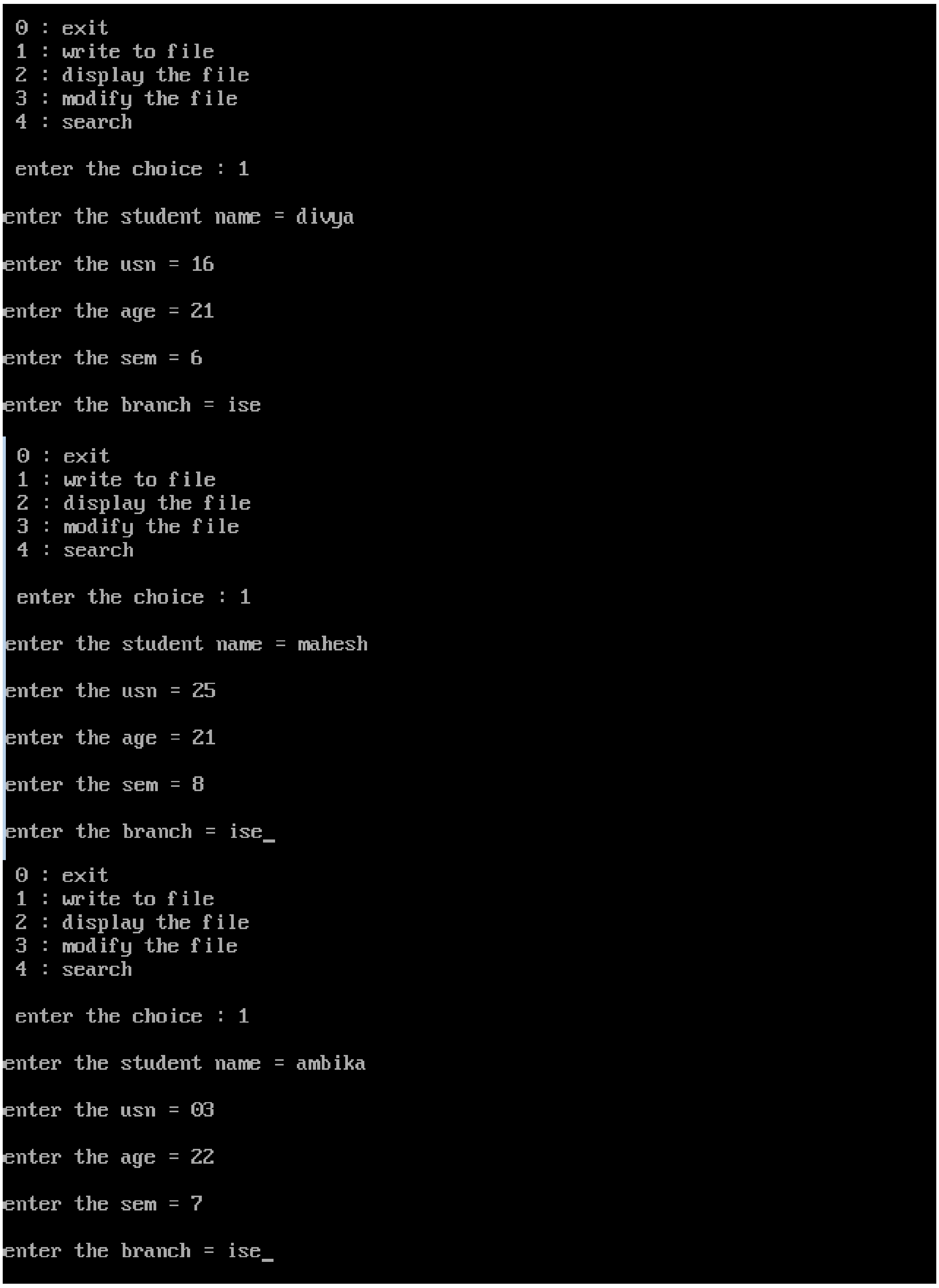
3. Write a C++ program to read and write student objects with Variable - Length records using any suitable record structure. Implement pack ( ), unpack ( ), modify ( ) and search ( ) methods.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<iostream.h>
#include<fstream.h>
#include<string.h>
#include<iomanip.h>

class student
{
        public: char name[10];
                char usn[10];
                char age[5];
                char sem[5];
                char branch[5];
                char buffer[100];
};

fstream file;
student s;

void writerecord()
{
        file.open("program_3.txt",ios::app);
        if(!file)
        {
                cout<<"cannot open the file in append mode";
                getch();
                exit(1);
        }
        cout<<"\nenter the student name = ";
        cin>>s.name;
        cout<<"\nenter the usn = ";
        cin>>s.usn;
        cout<<"\nenter the age = ";
        cin>>s.age;
        cout<<"\nenter the sem = ";
        cin>>s.sem;
        cout<<"\nenter the branch = ";
        cin>>s.branch;


        strcpy(s.buffer,s.name);
        strcat(s.buffer,"|");
        strcat(s.buffer,s.usn);
        strcat(s.buffer,"|");
        strcat(s.buffer,s.age);
        strcat(s.buffer,"|");
        strcat(s.buffer,s.sem);
        strcat(s.buffer,"|");
```

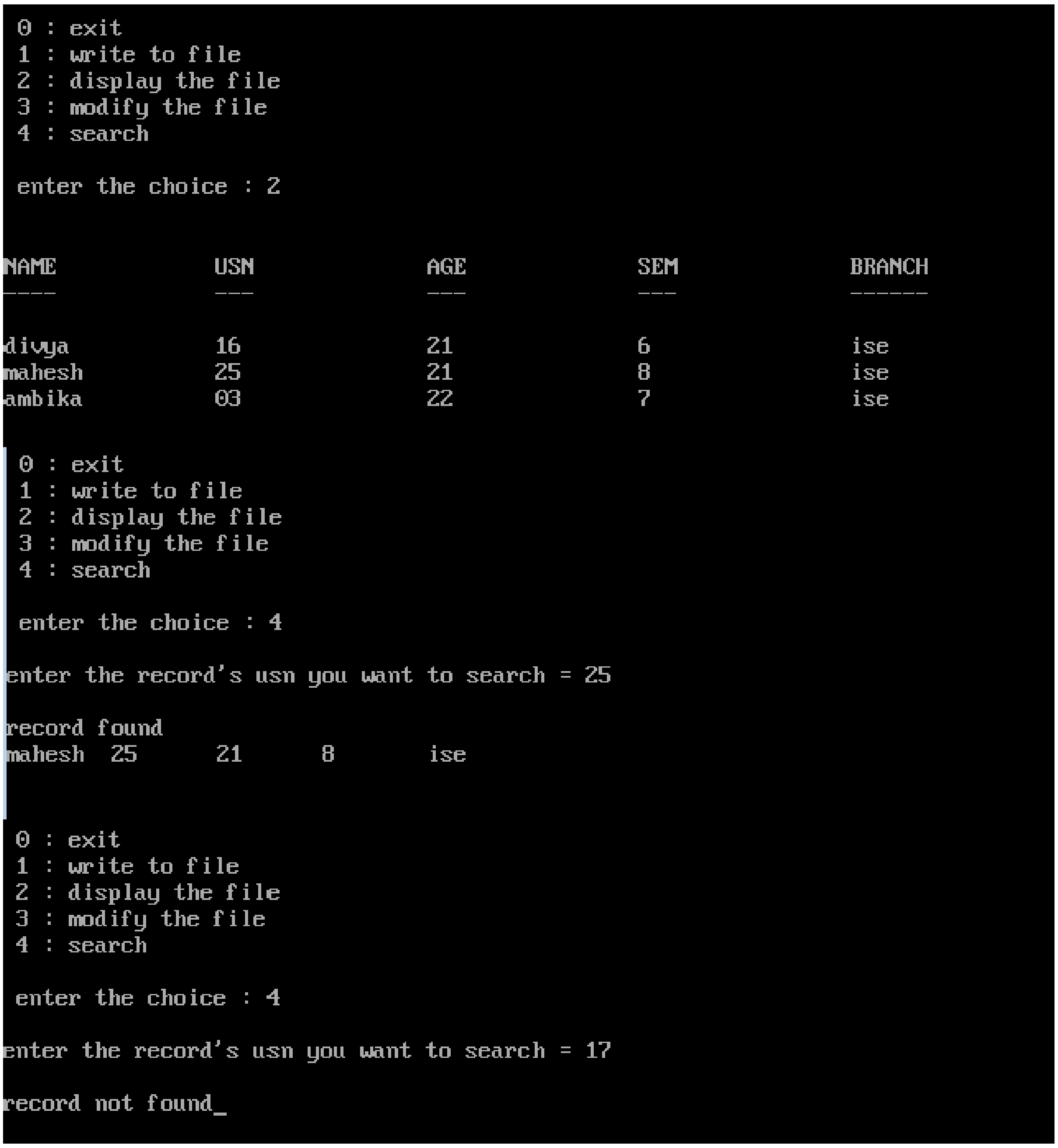
```
	strcat(s.buffer,s.branch);

	strcat(s.buffer,"\n");
	file<<s.buffer;
	file.close();
}

void search()
{
	char usn[10];
	char extra[45];

	file.open("program_3.txt",ios::in);
	if(!file)
	{
		cout<<"\nunable to open the file in read mode";
		getch();
		exit(0);
	}
	cout<<"\nenter the record's usn you want to search = ";
	cin>>usn;

	while(!file.eof())
	{
		file.getline(s.name,10,'|');
		file.getline(s.usn,10,'|');
		file.getline(s.age,5,'|');
		file.getline(s.sem,5,'|');
		file.getline(s.branch,5,'\n');


		if(strcmp(s.usn,usn)==0)
		{
			cout<<"\nrecord found";
			cout<<"\nname\tusn\tage\tsem\tbranch";

			cout<<"\n"<<s.name<<"\t"<<s.usn<<"\t";
			cout<<s.age<<"\t"<<s.sem<<"\t"<<s.branch;

			file.close();
			getch();
			return;
		}
	}
	cout<<"\nrecord not found";
	file.close();
	getch();
	return;
}

void displayFile()
{
	int i;
	file.open("program_3.txt",ios::in);
	if(!file)
	{
		cout<<"\ncannot open the file in read mode";
		getch();
		exit(1);
	}

	i=0;
	printf("\n\nNAME\t\tUSN\t\tAGE\t\tSEM\t\tBRANCH\n");


	while(!file.eof())
	{
		file.getline(s.name,15,'|');
		file.getline(s.usn,15,'|');
		file.getline(s.age,5,'|');
		file.getline(s.sem,5,'|');
		file.getline(s.branch,5,'\n');
		printf("\n%s\t\t%s\t\t%s\t\t%s\t\t%s",s.name,s.usn,s.age,s.sem,s.branch);
		i++;
	}
	file.close();
	getch();
}

void modify()
{
	char usn[10];

	int i;
	int j;
	student s[100];

	file.open("program_3.txt",ios::in);
	if(!file)
	{
		cout<<"\nunable to open the file in input mode";
		getch();
		exit(1);
	}
	cout<<"\nenter the usn ";
	cin>>usn;

	i=0;
	while(!file.eof())
	{
		file.getline(s[i].name,15,'|');
		file.getline(s[i].usn,15,'|');
		file.getline(s[i].age,5,'|');
		file.getline(s[i].sem,5,'|');
		file.getline(s[i].branch,5,'\n');
		i++;
	}
```

```
        i--;

        for(j=0;j<i;j++)
        {
                if(strcmp(usn,s[j].usn)==0)
                {
                        cout<<"\nthe old values of the record with usn"<<usn<<"are";
                        cout<<"\nname = "<<s[j].name;
                        cout<<"\nusn = "<<s[j].usn;
                        cout<<"\nage = "<<s[j].age;
                        cout<<"\nsem = "<<s[j].sem;
                        cout<<"\nbranch = "<<s[j].branch;

                        cout<<"\nenter the new values\n";
                        cout<<"\nname = ";
                        cin>>s[j].name;
                        cout<<"\nusn = ";
                        cin>>s[j].usn;
                        cout<<"\nage = ";
                        cin>>s[j].age;
                        cout<<"\nsem = ";
                        cin>>s[j].sem;
                        cout<<"\nbranch = ";
                        cin>>s[j].branch;
                        break;
                }
        }

        if(j==i)
        {
                cout<<"\nthe record with usn " <<usn<< "is not present ";
                getch();
                return;
        }
        file.close();

        file.open("program_3.txt",ios::out);
        if(!file)
        {
                cout<<"\nunable to open the file in output mode";
                getch();
                return;
        }

        for(j=0;j<i;j++)
        {
                file<<s[j].name<<'|'<<s[j].usn<<'|'<<s[j].age
                        <<'|'<<s[j].sem<<'|'<<s[j].branch<<'\n';
        }
        file.close();
}

void main()
{
        int choice;
        while(1)
        {
                clrscr();
                cout<<"\n 0 : exit";
                cout<<"\n 1 : write to file";
                cout<<"\n 2 : display the file";
                cout<<"\n 3 : modify the file";
                cout<<"\n 4 : search";
                cout<<"\n\n enter the choice : ";
                cin>>choice;

                switch(choice)
                {
                        case 1: writerecord();
                                break;

                        case 2: displayFile();
                                break;

                        case 3: modify();
                                break;

                        case 4: search();
                                break;

                        case 0: exit(0);

                        default: cout<<"\ninvalid input...";
                                break;
                }
        }
}
```

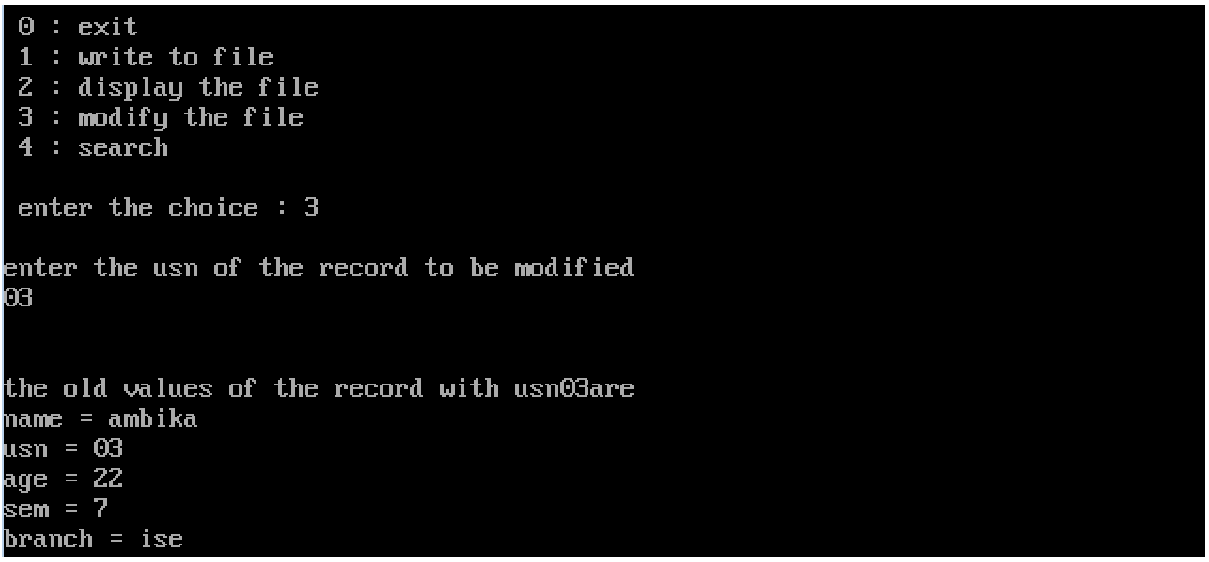
OUTPUT:



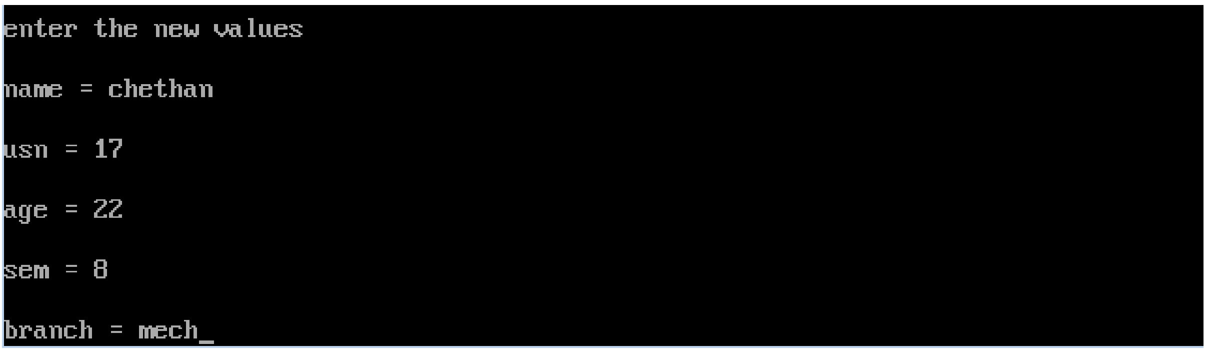
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 1

enter the student name = divya

enter the usn = 16

enter the age = 21

enter the sem = 6

enter the branch = ise

 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 1

enter the student name = mahesh

enter the usn = 25

enter the age = 21

enter the sem = 8

enter the branch = ise_
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 1

enter the student name = ambika

enter the usn = 03

enter the age = 22

enter the sem = 7

enter the branch = ise_
```



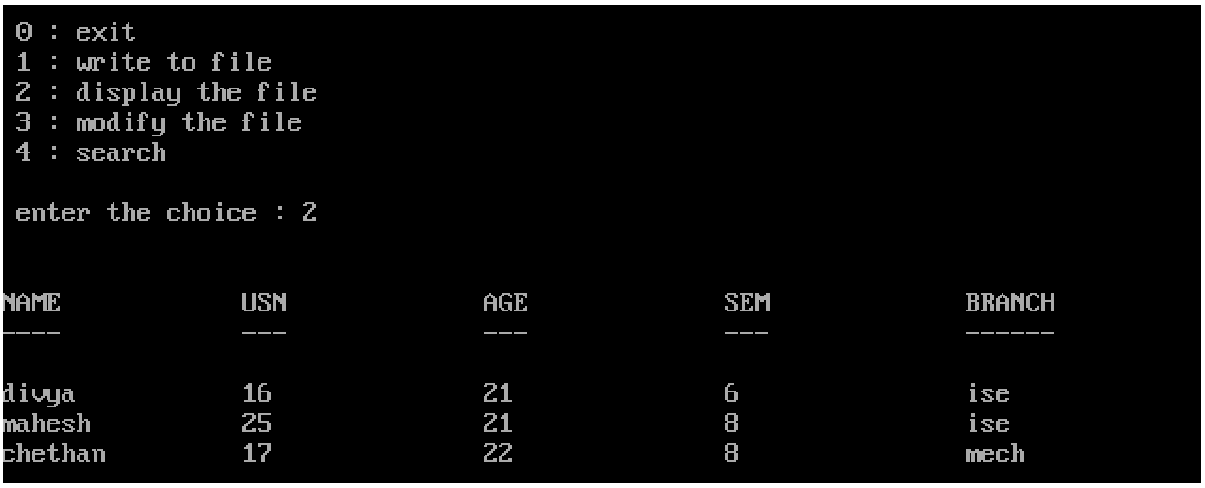
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 2


NAME            USN             AGE             SEM             BRANCH
----            ---             ---             ---             ------

divya           16              21              6               ise
mahesh          25              21              8               ise
ambika          03              22              7               ise

 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 4

enter the record's usn you want to search = 25

record found
mahesh  25      21      8       ise


 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 4

enter the record's usn you want to search = 17

record not found_
```



```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 3

enter the usn of the record to be modified
03


the old values of the record with usn03are
name = ambika
usn = 03
age = 22
sem = 7
branch = ise
```



```
enter the new values

name = chethan

usn = 17

age = 22

sem = 8

branch = mech_
```



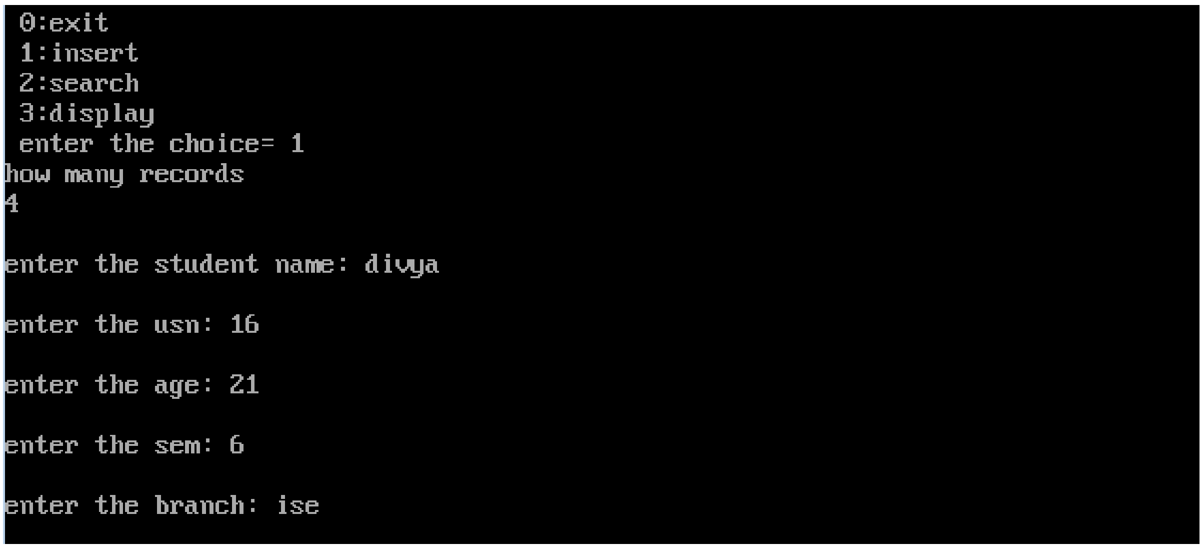
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 2


NAME            USN             AGE             SEM             BRANCH
----            ---             ---             ---             ------

divya           16              21              6               ise
mahesh          25              21              8               ise
chethan         17              22              8               mech
```

4. Write a C++ program to write student objects with Variable – Length records using any suitable record structure and to read from this file a student record using RRN.

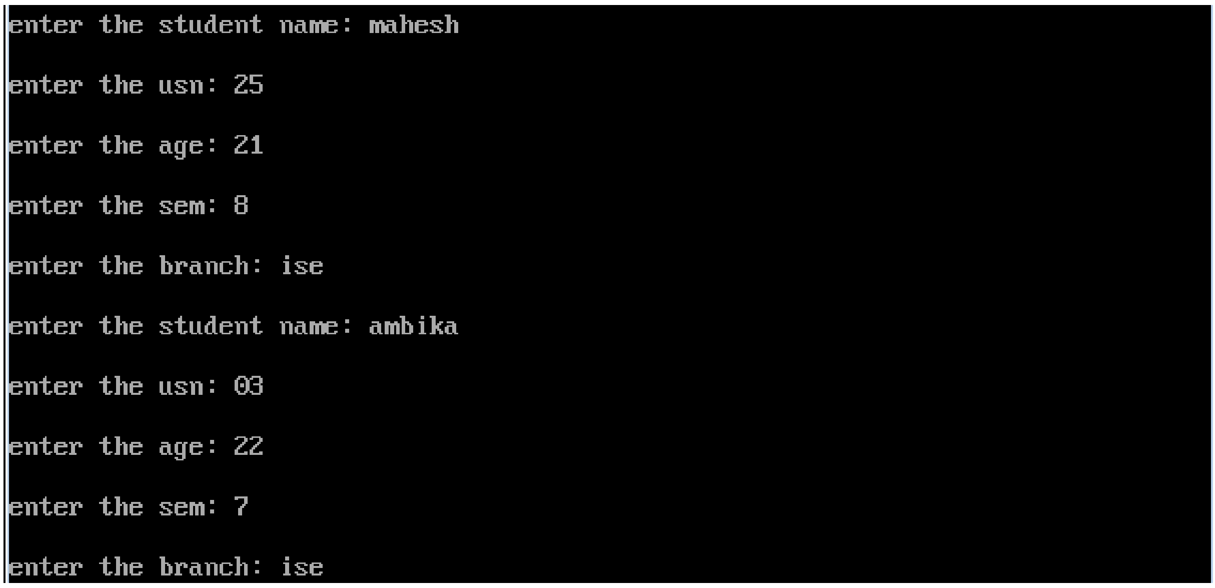
```
#include<stdio.h>
#include<stdlib.h>
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
#include<string.h>
#include<iomanip.h>

class student
{
        public:char name[15],usn[10],age[5],sem[5],branch[15],buffer[100];
};


void writerecord()
{
        fstream file;
        student s;
        int k,n;
        file.open("program_4.txt",ios::app);
        if(!file)
        {
                cout<<"\ncan not open the file in append mode\n";
                getch();
                exit(0);

        }
        printf("how many records\n");
        scanf("%d",&n);
        for(k=0;k<n;k++)
        {
                cout<<"\nenter the student name: ";
                cin>>s.name;
                cout<<"\nenter the usn: ";
                cin>>s.usn;
                cout<<"\nenter the age: ";
                cin>>s.age;
                cout<<"\nenter the sem: ";
                cin>>s.sem;
                cout<<"\nenter the branch: ";
                cin>>s.branch;
                file<<k<<"|"<<s.name<<"|"<<s.usn<<"|"
                <<s.age<<"|"<<s.sem<<"|"<<s.branch<<"\n";
        }
        file.close();
}

void displayfile()
{
        student s;
        char rrn[10];
```

```
	fstream file;
	file.open("program_4.txt",ios::in);
	if(!file)
	{
		cout<<"\ncannot open the file in input mode\n";
		getch();
		exit(1);
	}
	cout<<"\n";
	printf("rrn\tname\t\tusn\t\tage\t\tsem\t\tbranch\n");
	while(!file.eof())
	{
		file.getline(rrn,4,'|');
		file.getline(s.name,15,'|');
		file.getline(s.usn,15,'|');
		file.getline(s.age,5,'|');
		file.getline(s.sem,5,'|');
		file.getline(s.branch,15,'\n');
		printf("\n%s\t%s\t\t%s\t\t%s\t\t%s\t\t%s\n",
		rrn,s.name,s.usn,s.age,s.sem,s.branch);

	}
	file.close();
	getch();
}

void search()
{
	char rrn[10],rrn1[10][15];
	int i;
	student std[100];
	cout<<"\n enter the rrn to be searched";
	cin>>rrn;
	fstream file;
	file.open("program_4.txt",ios::in);
	if(!file)
	{
		cout<<"\n can not open the file in input mode";
		getch();
		exit(0);
	}
	i=0;
	printf("\n rrn\tname\tusn\tage\tsem\tbranch\n");
	while(!file.eof())
	{
		file.getline(rrn1[i],4,'|');
		file.getline(std[i].name,15,'|');
		file.getline(std[i].usn,15,'|');
		file.getline(std[i].age,5,'|');
		file.getline(std[i].sem,5,'|');
		file.getline(std[i].branch,15,'\n');
		i++;
	}
	for(int j=0;j<i-1;j++)
	{
		if(strcmp(rrn,rrn1[j])==0)
		{
			printf("\n%s\t%s\t%s\t%s\t%s\t%s\n",
			rrn,std[j].name,std[j].usn,std[j].age,
			std[j].sem,std[j].branch);
			printf("\n record found\n");
			file.close();
			return;
		}
	}
	cout<<"\nrecord not found\n";
	file.close();
	return;
}

void main()
{
	int choice;
	clrscr();
	while(1)
	{
		cout<<"\n 0:exit";
		cout<<"\n 1:insert";
		cout<<"\n 2:search";
		cout<<"\n 3:display";
		cout<<"\n enter the choice=";
		cin>>choice;
		switch(choice)
		{
			case 1:writerecord();
				break;
			case 2:search();
				break;
			case 3:displayfile();
				break;
			case 0:exit(0);
			default:cout<<"\n invalid option";
				break;
		}

	}
}
```

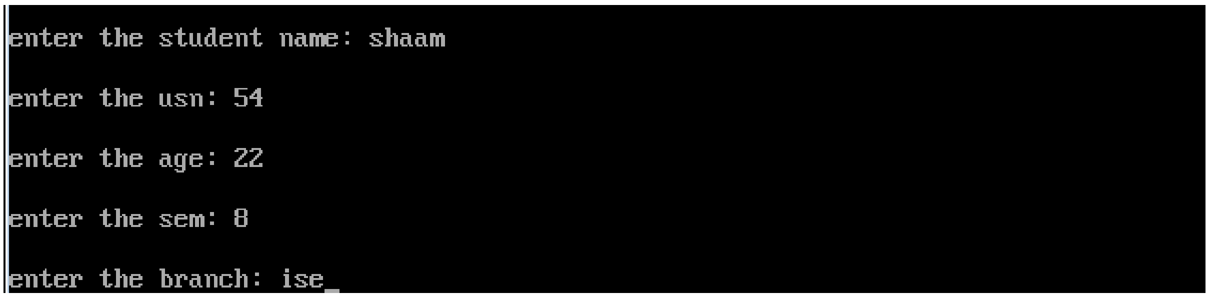
OUTPUT:



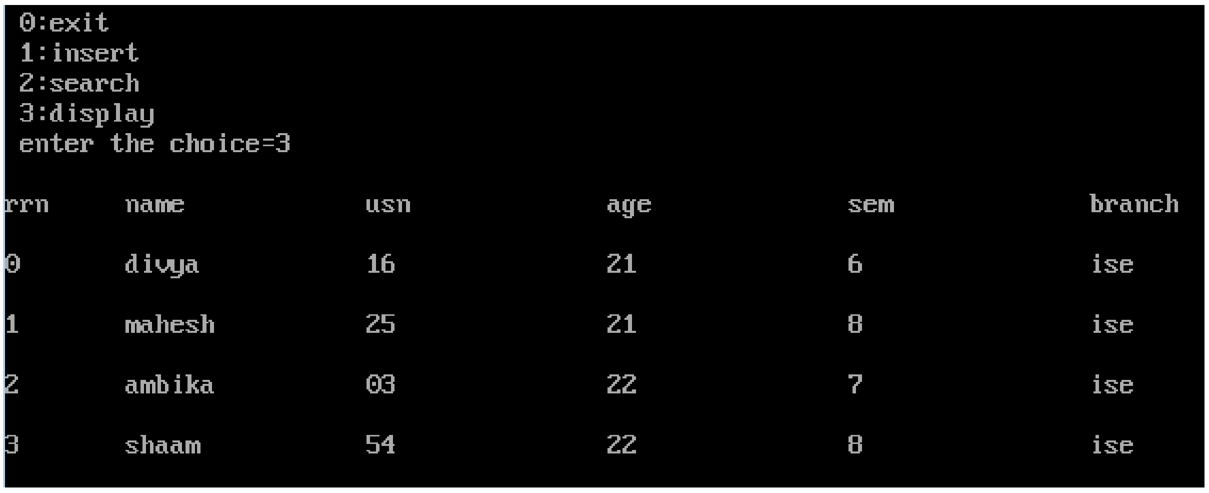
```
 0:exit
 1:insert
 2:search
 3:display
 enter the choice= 1
how many records
4

enter the student name: divya

enter the usn: 16

enter the age: 21

enter the sem: 6

enter the branch: ise
```



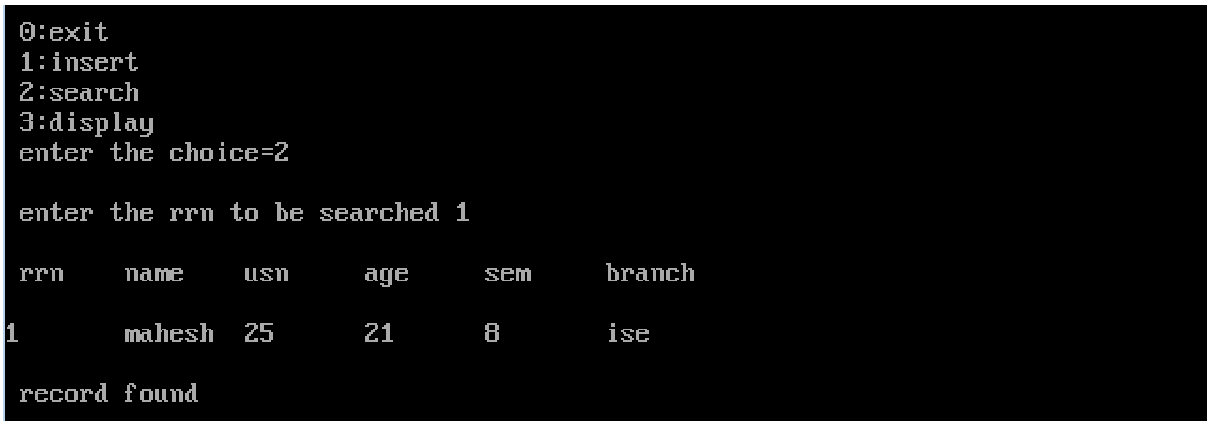
```
enter the student name: mahesh

enter the usn: 25

enter the age: 21

enter the sem: 8

enter the branch: ise

enter the student name: ambika

enter the usn: 03

enter the age: 22

enter the sem: 7

enter the branch: ise
```



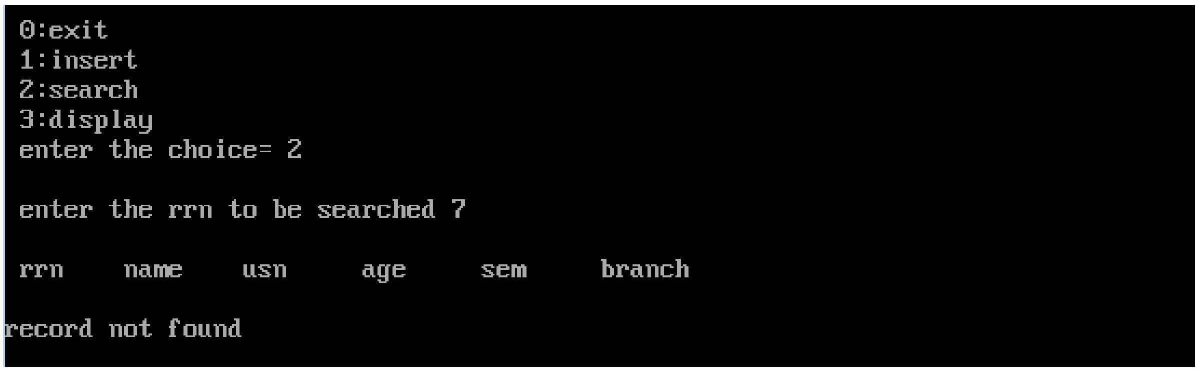
```
enter the student name: shaam

enter the usn: 54

enter the age: 22

enter the sem: 8

enter the branch: ise_
```



```
 0:exit
 1:insert
 2:search
 3:display
 enter the choice=3

rrn     name            usn             age             sem             branch

0       divya           16              21              6               ise

1       mahesh          25              21              8               ise

2       ambika          03              22              7               ise

3       shaam           54              22              8               ise
```



```
 0:exit
 1:insert
 2:search
 3:display
 enter the choice=2

 enter the rrn to be searched 1

 rrn    name    usn     age     sem     branch

1       mahesh  25      21      8       ise

 record found
```



```
 0:exit
 1:insert
 2:search
 3:display
 enter the choice= 2

 enter the rrn to be searched 7

 rrn    name    usn     age     sem     branch

record not found
```

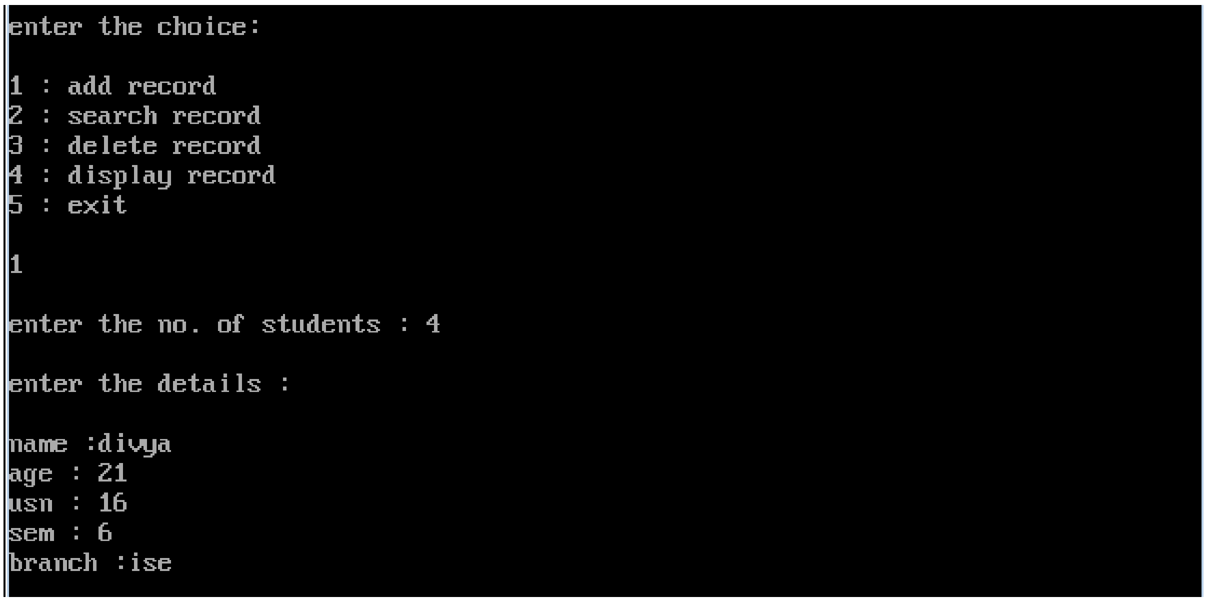
5. Write a C++ program to implement simple index on primary key for a file of student objects. Implement add ( ), search ( ), delete ( ) using the index.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
#include<fstream.h>
#include<iostream.h>
#include<iomanip.h>

class record
{
        public: char age[5];
                char usn[20];
                char name[20];
                char sem[2];
                char branch[5];
}rec[20];

char st_no[5];
int no;

void retrieve_details()
{
        fstream file2;
        char name[20];
        char age[5];
        char usn[20];
        char sem[2];
        char branch[5];
        char ind[5];

        file2.open("record.txt",ios::in);

        for(int i=0;i<no;i++)
        {
                file2.getline(ind,5,'|');
                file2.getline(usn,20,'|');
                file2.getline(name,20,'|');
                file2.getline(age,5,'|');
                file2.getline(sem,5,'|');
                file2.getline(branch,5,'\n');

                if(strcmp(ind,st_no)==0)
                {
                        cout<<"\n\t"<<"student details :\n";
                        cout<<"\n\tUSN\tNAME\tAGE\tSEM\tBRANCH\n";
                        cout<<"\n\t"<<usn<<"\t"<<name<<"\t";
                        cout<<age<<"\t"<<sem<<"\t"<<branch<<"\n";
                }
        }
        file2.close();
}
```

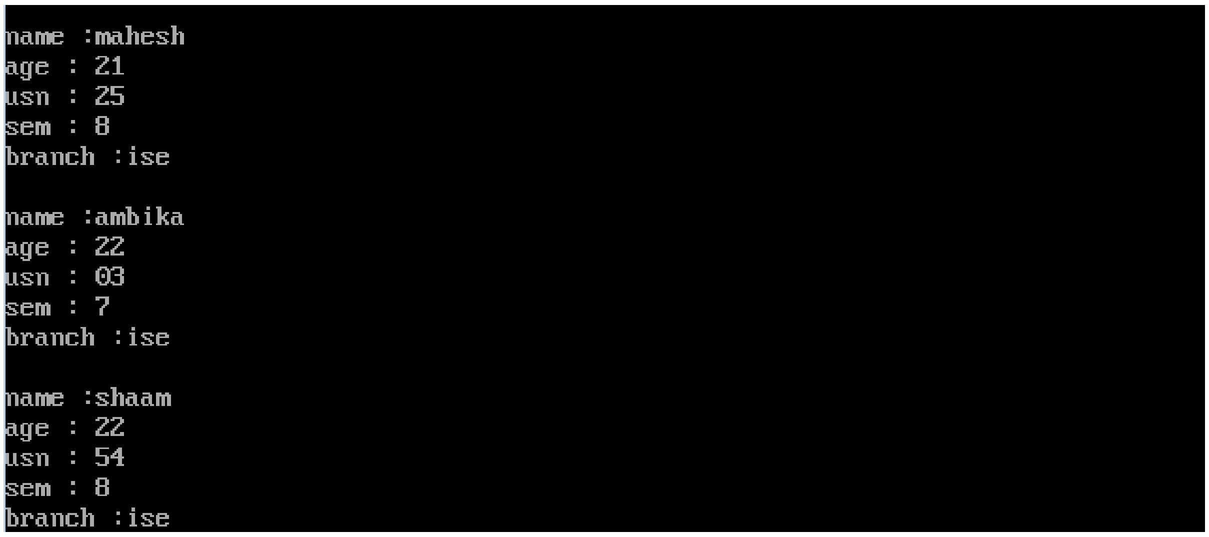
```
void delete_record(char usno[])
{
        int i;
        fstream file1,file2;
        char name[20];
        char age[5];
        char usn[20];
        char sem[2];
        char branch[5];
        char ind[5];

        file2.open("record.txt",ios::in);

        for(i=0;i<no;i++)
        {
                file2.getline(ind,5,'|');
                file2.getline(usn,20,'|');
                file2.getline(name,20,'|');
                file2.getline(age,5,'|');
                file2.getline(sem,5,'|');
                file2.getline(branch,5,'\n');

                strcpy(rec[i].usn,usn);
                strcpy(rec[i].name,name);
                strcpy(rec[i].age,age);
                strcpy(rec[i].sem,sem);
                strcpy(rec[i].branch,branch);
        }

        int flag=-1;

        for(i=0;i<no;i++)
        {
                if(strcmp(rec[i].usn,usno)==0)
                {
                        flag=i;
                }
        }

        if(flag==-1)
        {
                cout<<"error..! \n";
                return;
        }

        if(flag==(no-1))
        {
                no--;
                cout<<"record deleted !\n";
                return;
        }

        for(i=flag;i<no;i++)
        {
                rec[i]=rec[i+1];
        }
        no--;
        cout<<"\nrecord deleted !\n";
        file2.close();

        file1.open("index.txt",ios::out);
        file2.open("record.txt",ios::out);

        for(i=0;i<no;i++)
        {
                file1<<rec[i].usn<<"|"<<i<<"\n";
                file2<<i<<"|"<<rec[i].usn<<"|"<<rec[i].name<<"|"<<rec[i].age
                        <<"|"<<rec[i].sem<<"|"<<rec[i].branch<<"\n";
        }

        file1.close();
        file2.close();
        return;
}

int main()
{
        fstream file1,file2;
        int choice;
        char rt_usn[20];
        char st_usn[20];
        char ind[2];
        char name[20];
        char age[2];
        char sem[5];
        char branch[5];
        int i;
        int flag;
        int flag1;
        clrscr();

        file1.open("index.txt",ios::out);
        file2.open("record.txt",ios::out);

        if(!file1 || !file2)
        {
                cout<<"file creation error ! \n";
                exit(0);
        }

        for(;;)
        {
                cout<<"\nenter the choice:\n\n";
                cout<<"1 : add record\n";
                cout<<"2 : search record\n";
                cout<<"3 : delete record\n";
                cout<<"4 : display record\n";
        cout<<"5 : exit\n\n";

        cin>>choice;

        switch(choice)
        {
                case 1: cout<<"\nenter the no. of students : ";
                        cin>>no;
                        cout<<"\nenter the details :\n";
                        for(i=0;i<no;i++)
                        {
                                cout<<"\nname :";
                                cin>>rec[i].name;
                                cout<<"age : ";
                                cin>>rec[i].age;
                                cout<<"usn : ";
                                cin>>rec[i].usn;
                                cout<<"sem : ";
                                cin>>rec[i].sem;
                                cout<<"branch :";
                                cin>>rec[i].branch;


                                file1<<rec[i].usn<<"|"<<i<<"\n";
                                file2<<i<<"|"<<rec[i].usn<<"|"<<rec[i].name
                                        <<"|"<<rec[i].age<<"|"<<rec[i].sem
                                        <<"|"<<rec[i].branch<<"\n";
                        }

                        file1.close();
                        file2.close();
                        break;

                case 2: cout<<"\nenter the USN of the student record to be searched\n";
                        cin>>st_usn;
                        file1.open("index.txt",ios::in);
                        if(!file1)
                        {
                                cout<<"error!\n";
                                exit(0);
                        }
                        flag1=0;
                        for(i=0;i<no;i++)
                        {
                                file1.getline(rt_usn,20,'|');
                                file1.getline(st_no,4,'\n');
                                if(strcmp(st_usn,rt_usn)==0)
                                {
                                        retrieve_details();
                                        flag1=1;
                                }
                        }
                        if(!flag1)
                        {
```

```
                                        cout<<"\nrecord search failed !!\n";
                                }
                                file1.close();
                                break;

                        case 3: cout<<"\nenter the USN of the student record to be deleted\n\n";
                                cin>>st_usn;
                                file1.open("index.txt",ios::in);
                                if(!file1)
                                {
                                        cout<<"error !\n";
                                        exit(0);
                                }
                                flag=0;
                                for(i=0;i<no;i++)
                                {
                                        file1.getline(rt_usn,20,'|');
                                        file1.getline(st_no,4,'\n');
                                        if(strcmp(st_usn,rt_usn)==0)
                                        {
                                                delete_record(rt_usn);
                                                flag=1;
                                        }
                                }
                                if(!flag)
                                {
                                        cout<<"deletion failed!\n";
                                }
                                file1.close();
                                break;

                        case 4: cout<<"\n\tUSN\tNAME\tAGE\tSEM\tBRANCH\t\n";
                                for(i=0;i<no;i++)
                                {
                                        cout<<"\n\t"<<rec[i].usn;
                                        cout<<"\t"<<rec[i].name;
                                        cout<<"\t"<<rec[i].age;
                                        cout<<"\t"<<rec[i].sem;
                                        cout<<"\t"<<rec[i].branch<<"\n";
                                }
                                break;

                        case 5: exit(0);

                        default: cout<<"invalid choice !\n";
                                break;
                }
        }
}
```

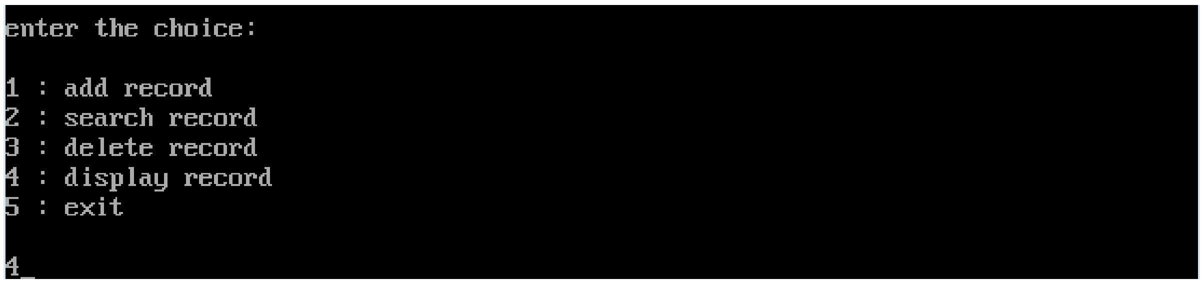
OUTPUT:



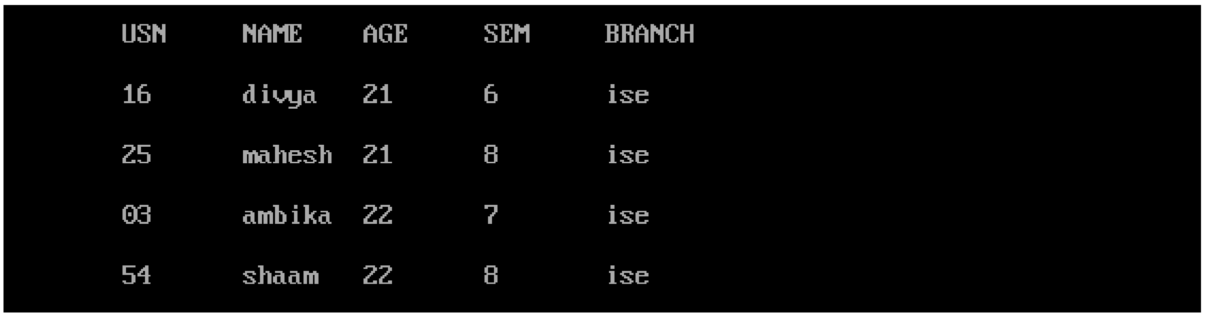
```
enter the choice:

1 : add record
2 : search record
3 : delete record
4 : display record
5 : exit

1

enter the no. of students : 4

enter the details :

name :divya
age : 21
usn : 16
sem : 6
branch :ise
```



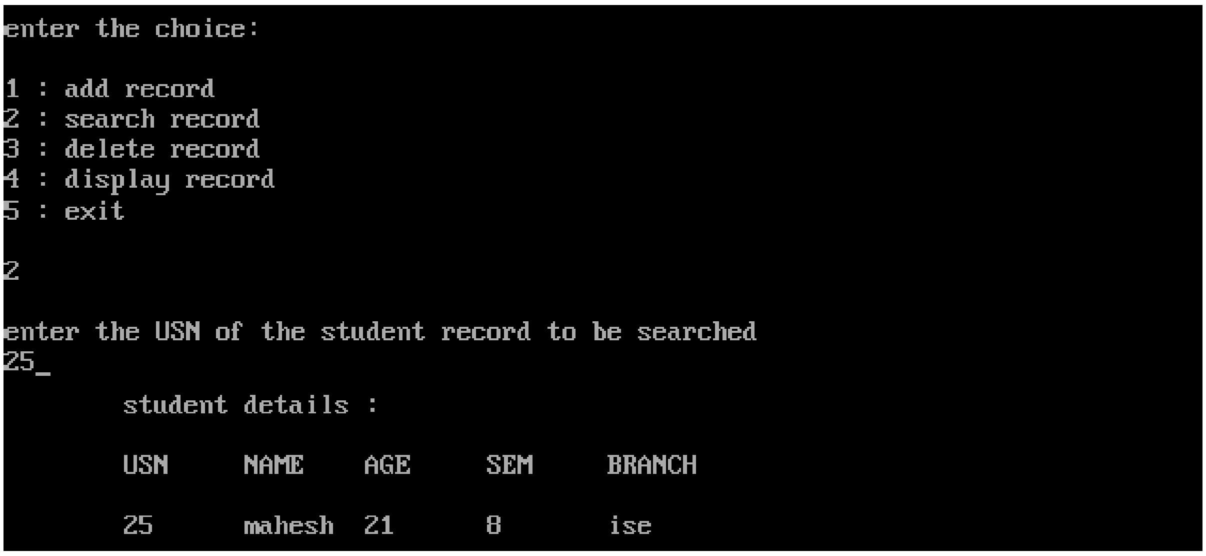
```
name :mahesh
age : 21
usn : 25
sem : 8
branch :ise

name :ambika
age : 22
usn : 03
sem : 7
branch :ise

name :shaam
age : 22
usn : 54
sem : 8
branch :ise
```



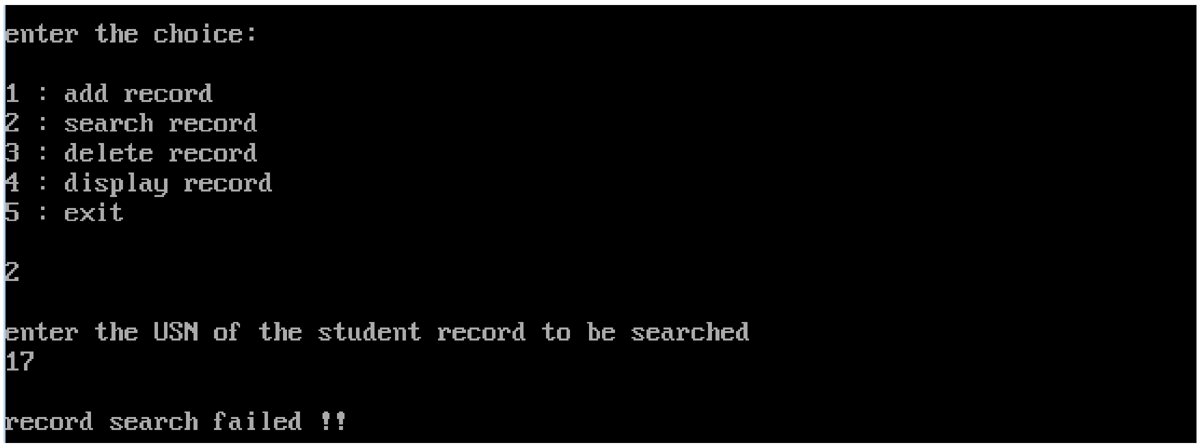
```
enter the choice:

1 : add record
2 : search record
3 : delete record
4 : display record
5 : exit

4_
```



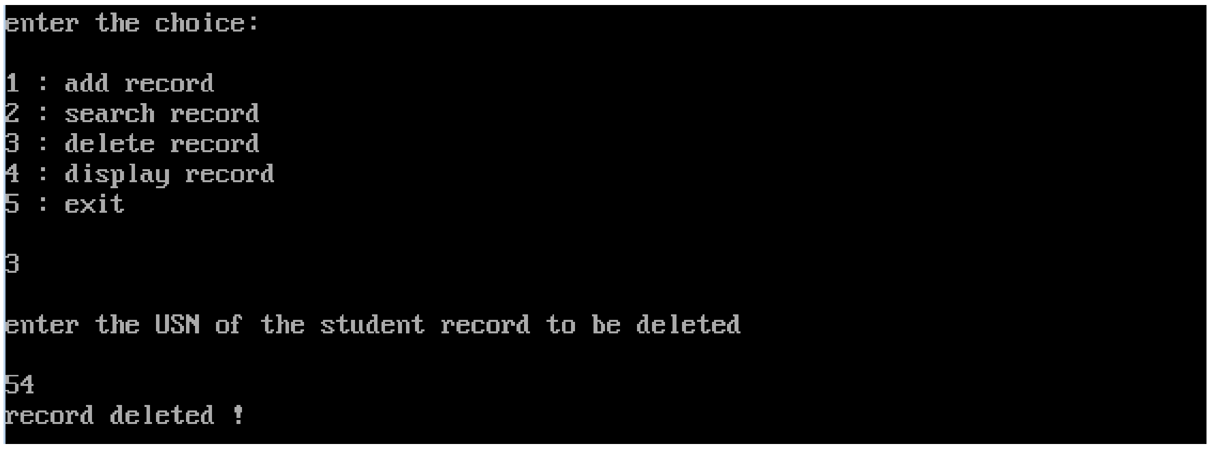
```
        USN     NAME    AGE     SEM     BRANCH

        16      divya   21      6       ise

        25      mahesh  21      8       ise

        03      ambika  22      7       ise

        54      shaam   22      8       ise
```



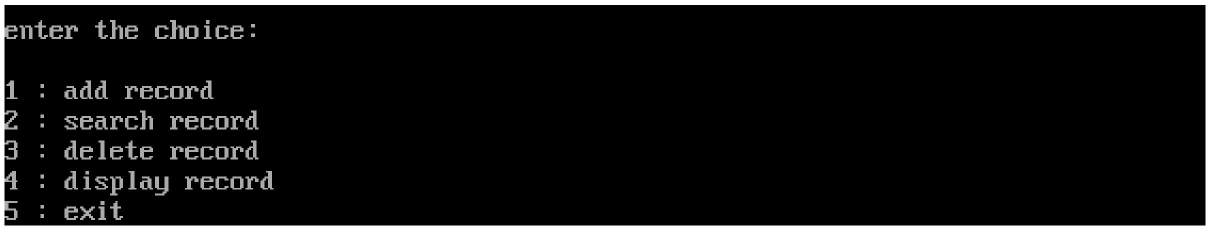
```
enter the choice:

1 : add record
2 : search record
3 : delete record
4 : display record
5 : exit

2

enter the USN of the student record to be searched
25_
        student details :

        USN     NAME    AGE     SEM     BRANCH

        25      mahesh  21      8       ise
```



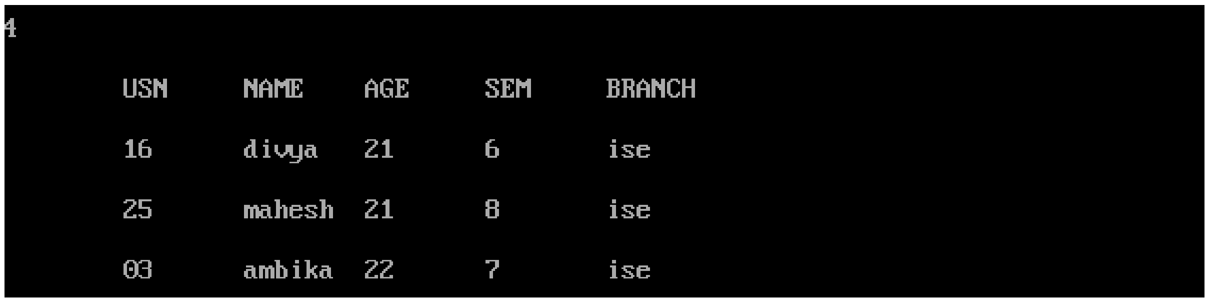
```
enter the choice:

1 : add record
2 : search record
3 : delete record
4 : display record
5 : exit

2

enter the USN of the student record to be searched
17

record search failed !!
```



```
enter the choice:

1 : add record
2 : search record
3 : delete record
4 : display record
5 : exit

3

enter the USN of the student record to be deleted

54
record deleted !
```



```
enter the choice:

1 : add record
2 : search record
3 : delete record
4 : display record
5 : exit
```



```
4

        USN     NAME    AGE     SEM     BRANCH

        16      divya   21      6       ise

        25      mahesh  21      8       ise

        03      ambika  22      7       ise
```

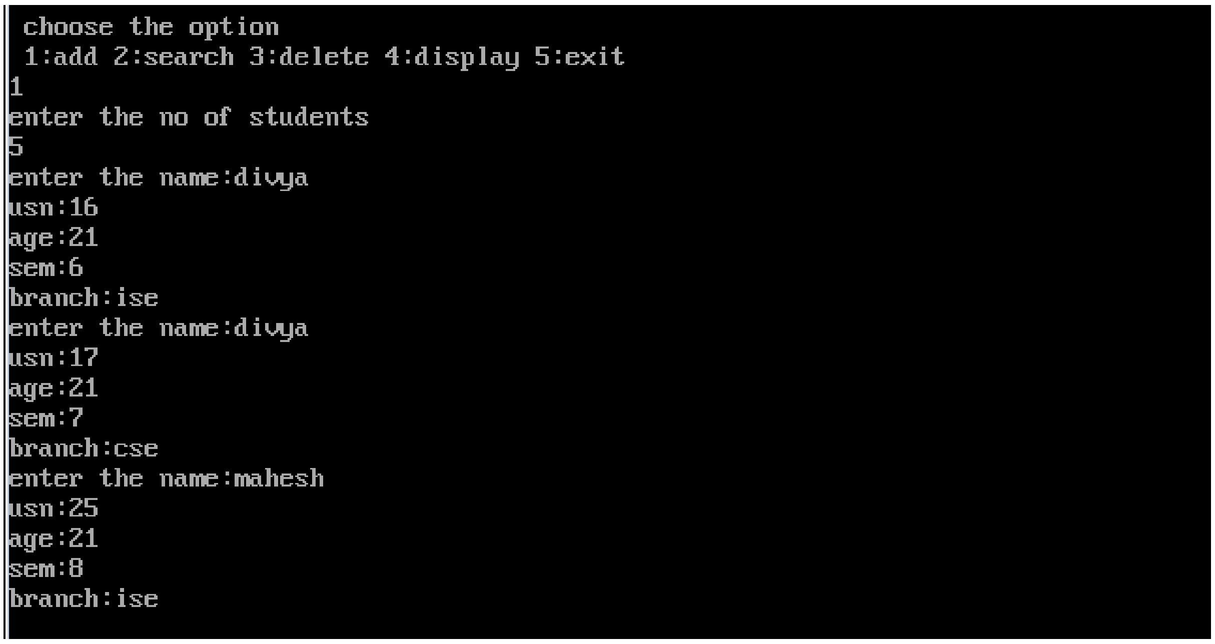
6. Write a C++ program to implement index on secondary key, the name, for a file of student objects. Implement add ( ), search ( ), delete ( ) using the secondary index.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<fstream.h>
#include<iostream.h>
#include<stdlib.h>

class record
{
        public:
                char age[5],sem[5],usn[20],name[20],branch[5];
}rec[20],found[20];

int no;
char st_no[5],rt_name[20];

void sortrecord()
{
        int i,j;
        record temp;
        for(i=0;i<no-1;i++)
        for(j=0;j<no-i-1;j++)
        if(strcmp(rec[j].name,rec[j+1].name)>0)
        {
                temp=rec[j];
                rec[j]=rec[j+1];
                rec[j+1]=temp;
        }
}

void indexfile()
{
        fstream index,index2;
        int i;
        index.open("secindex.txt",ios::out);
        index2.open("record.txt",ios::out);
        for(i=0;i<no;i++)
        {
                index<<rec[i].name<<"|"<<rec[i].usn<<"|"<<i<<"\n";
                index2<<i<<"|"<<rec[i].name<<"|"<<rec[i].usn<<"|"<<
                        rec[i].age<<"|"<<rec[i].sem<<"|"<<rec[i].branch<<"\n";
        }
        index.close();
        index2.close();
}

void retrieve_record(char *index)
{
        fstream file;
        char age[5],sem[5],usn[20],name[20],branch[5],ind[5];
        file.open("record.txt",ios::in);
        for(int i=0;i<no;i++)
        {
                file.getline(ind,5,'|');
                file.getline(name,20,'|');
                file.getline(usn,20,'|');
                file.getline(age,5,'|');
                file.getline(sem,5,'|');
                file.getline(branch,5,'\n');
                if(strcmp(index,ind)==0)
                {
                        cout<<"USN\tNAME\tAGE\tSEM\tBRANCH\n";
                        cout<<usn<<"\t"<<name<<"\t"<<age<<"\t"<<sem<<"\t"<<branch<<"\n";
                }
        }
        file.close();
}

void retrieve_details()
{
        fstream file;
        char age[5],sem[5],usn[20],name[20],branch[5],ind[5];
        char chusn[20],index[20][20];
        file.open("secindex.txt",ios::in);
        int k=0;
        for(int i=0;i<no;i++)
        {
                file.getline(name,20,'|');
                file.getline(usn,20,'|');
                file.getline(ind,4,'\n');
                if(strcmp(name,rt_name)==0)
                {
                        strcpy(found[k].name,name);
                        strcpy(found[k].usn,usn);
                        strcpy(index[k],ind);
                        k++;
                }

        }
        file.close();
        if(k==1)
        {
                retrieve_record(index[0]);
                return;
        }
        else
        {
        cout<<"choose the candidates usn\n";
        for(i=0;i<k;i++)
        cout<<"USN:"<<found[i].usn<<"\tNAME:"<<found[i].name<<endl;
        }
        cin>>chusn;
        for(i=0;i<k;i++)
        {
                if(strcmp(chusn,found[i].usn)==0)
                {
                        retrieve_record(index[i]);
                        return;
                }
        }
        cout<<"invalid entry\n";
        return;
}

void delete_record(char *indx)
{
        char age[5],sem[5],usn[20],name[20],branch[5],ind[5];
        fstream file1,file2;
        char index[20][20];
        file2.open("record.txt",ios::in);
        for(int i=0;i<no;i++)
        {
                file2.getline(ind,4,'|');
                file2.getline(name,20,'|');
                file2.getline(usn,20,'|');
                file2.getline(age,5,'|');
                file2.getline(sem,5,'|');
                file2.getline(branch,5,'\n');
                strcpy(index[i],ind);
                strcpy(rec[i].usn,usn);
                strcpy(rec[i].name,name);
                strcpy(rec[i].age,age);
                strcpy(rec[i].sem,sem);
                strcpy(rec[i].branch,branch);
        }
        int flag=-1;
        for(i=0;i<no;i++)
        {
                if(strcmp(index[i],indx)==0)
                flag=i;
        }
        if(flag==-1)
        {
                cout<<"error\n";
                return;
        }
        if(flag==(no-1))
        {
                no--;
                cout<<"record deleted\n";
                return;
        }
        for(i=flag;i<no;i++)
        {
                rec[i]=rec[i+1];
        }
        no--;
        cout<<"record deleted\n";
        file2.close();
        file1.open("secindex.txt",ios::in);
        file2.open("record.txt",ios::in);
        for(i=0;i<no;i++)
        {
                file1<<rec[i].name<<"|"<<rec[i].usn<<"|"<<i<<"\n";
                file2<<i<<"|"<<rec[i].name<<"|"<<rec[i].usn<<"|"<<
                        rec[i].age<<"|"<<rec[i].sem<<"|"<<rec[i].branch<<"\n";
        }
        file1.close();
        file2.close();
}

void delete_index(char *nam)
{
        fstream file;
        char age[5],sem[5],usn[20],name[20],branch[5],ind[5];
        char chusn[20],index[20][20];
        int i,k=0;
        file.open("secindex.txt",ios::in);
        for(i=0;i<no;i++)
        {
                file.getline(name,20,'|');
                file.getline(usn,20,'|');
                file.getline(ind,4,'\n');
                if(strcmp(nam,name)==0)
                {
                        strcpy(found[k].name,name);
                        strcpy(found[k].usn,usn);
                        strcpy(index[k],ind);
                        k++;
                }

        }
        file.close();
        if(k==1)
        {
                delete_record(index[0]);
                return;
        }
        else
        {
        cout<<"choose the candidates usn\n";
        for(i=0;i<k;i++)
        cout<<"USN:"<<found[i].usn<<" NAME:"<<found[i].name<<endl;
        }
        cin>>chusn;
        for(i=0;i<k;i++)
        {
                if(strcmp(chusn,found[i].usn)==0)
                {
                        delete_record(index[i]);
                        return;
                }
```

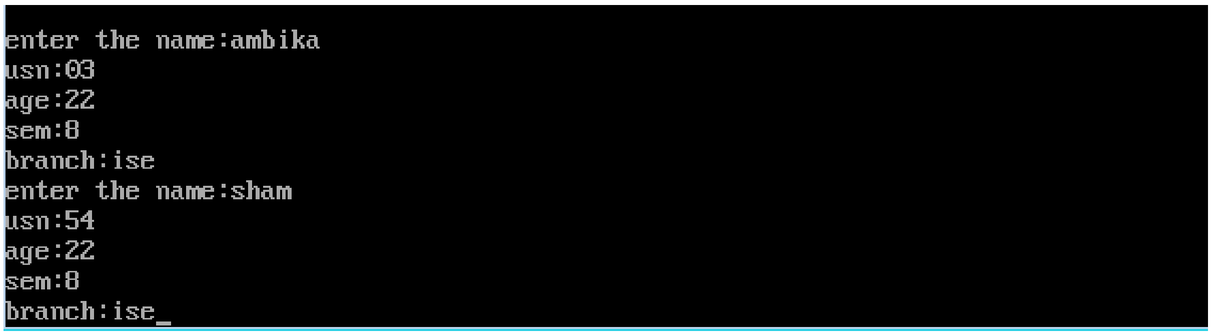
```
	}
	cout<<"invalid entry\n";
	return;
}


int main()
{
	fstream file1,file2;
	char rt_usn[20],st_name[20],st_usn[20];
	char age[5],sem[5],name[20],branch[5],ind[5];
	int i,flag,flag1,choice;
	clrscr();
	for(;;)
	{
		cout<<"\n choose the option\n 1:add 2:search 3:delete 4:display 5:exit\n";
		cin>>choice;
		switch(choice)
		{
			case 1:cout<<"enter the no of students\n";
				cin>>no;
				for(i=0;i<no;i++)
				{
					cout<<"enter the name:";
					cin>>rec[i].name;
					cout<<"usn:";
					cin>>rec[i].usn;
					cout<<"age:";
					cin>>rec[i].age;
					cout<<"sem:";
					cin>>rec[i].sem;
					cout<<"branch:";
					cin>>rec[i].branch;
				}
				sortrecord();
				indexfile();
				break;

			case 2: cout<<"enter the name of the record to be searched\n";
				cin>>st_name;
				file1.open("secindex.txt",ios::in);
				if(!file1)
				{
					cout<<"file creation error\n";
					exit(0);
				}
				flag1=0;
				for(i=0;i<no;i++)
				{
					file1.getline(rt_name,20,'|');
					file1.getline(st_usn,20,'|');
					file1.getline(st_no,4,'\n');
					if(strcmp(st_name,rt_name)==0)
					{
```

```
                                        retrieve_details();
                                        flag1=1;
                                }
                        }
                        if(!flag1)
                        cout<<"record search failed \n";
                        file1.close();
                        break;

                case 3: cout<<"enter the name of the record to be deleted\n";
                        cin>>st_name;
                        file1.open("secindex.txt",ios::in);
                        if(!file1)
                        {
                                cout<<"file creation error\n";
                                exit(0);
                        }
                        flag=0;
                        for(i=0;i<no;i++)
                        {
                                file1.getline(rt_name,20,'|');
                                file1.getline(st_usn,20,'|');
                                file1.getline(ind,4,'\n');
                                if(strcmp(st_name,rt_name)==0)
                                {
                                        delete_index(rt_name);
                                        flag=1;
                                        break;
                                }
                        }
                        if(!flag)
                        cout<<"deletion failed \n";
                        file1.close();
                        break;

                 case 4: cout<<"USN\tNAME\tAGE\tSEM\tBRANCH\n";
                        for(i=0;i<no;i++)
                        {
                        cout<<rec[i].usn<<"\t"<<rec[i].name<<"\t"<<rec[i].age<<"\t"
                        <<rec[i].sem<<"\t"<<rec[i].branch<<"\n";
                        }
                        break;

                default:cout<<"invalid choice\n";
                        exit(0);
                        break;
        }
    }
}
```

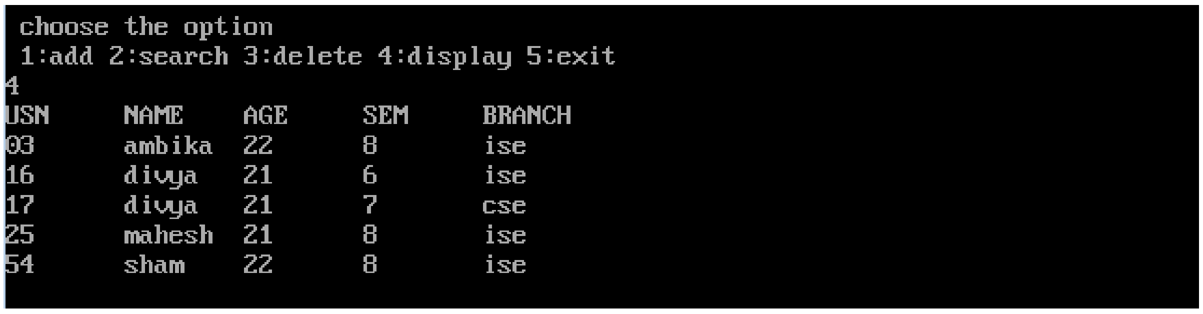
OUTPUT:



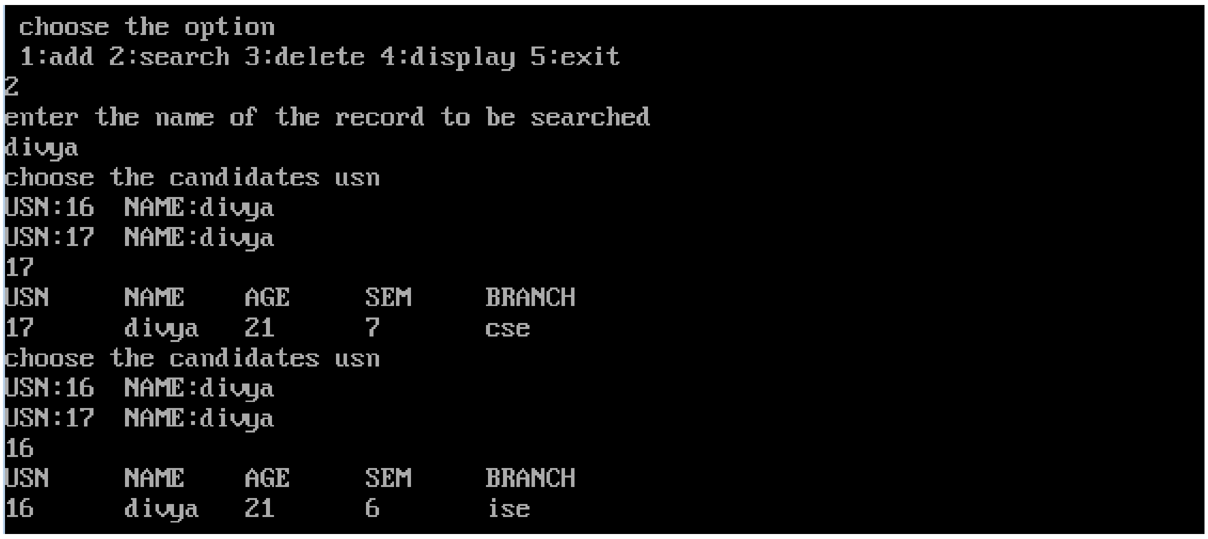
```
 choose the option
 1:add 2:search 3:delete 4:display 5:exit
1
enter the no of students
5
enter the name:divya
usn:16
age:21
sem:6
branch:ise
enter the name:divya
usn:17
age:21
sem:7
branch:cse
enter the name:mahesh
usn:25
age:21
sem:8
branch:ise
```



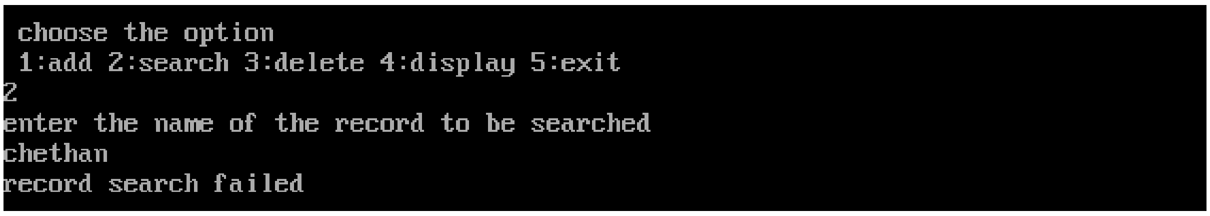
```
enter the name:ambika
usn:03
age:22
sem:8
branch:ise
enter the name:sham
usn:54
age:22
sem:8
branch:ise_
```



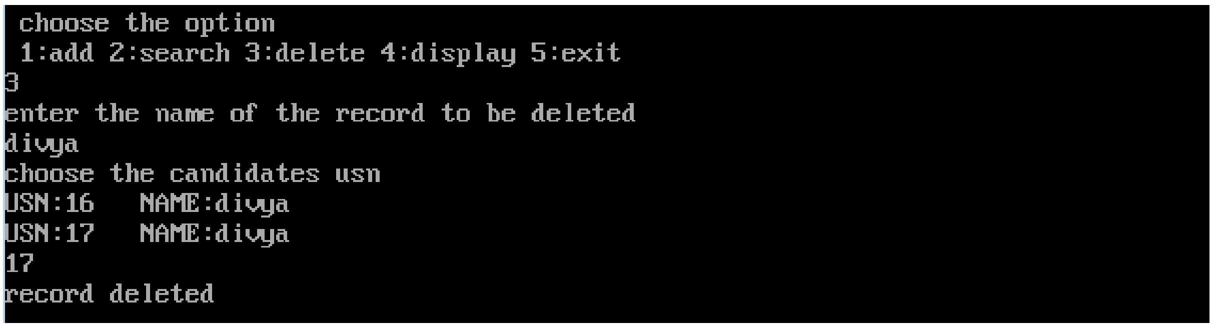
```
 choose the option
 1:add 2:search 3:delete 4:display 5:exit
4
USN     NAME    AGE     SEM     BRANCH
03      ambika  22      8       ise
16      divya   21      6       ise
17      divya   21      7       cse
25      mahesh  21      8       ise
54      sham    22      8       ise
```



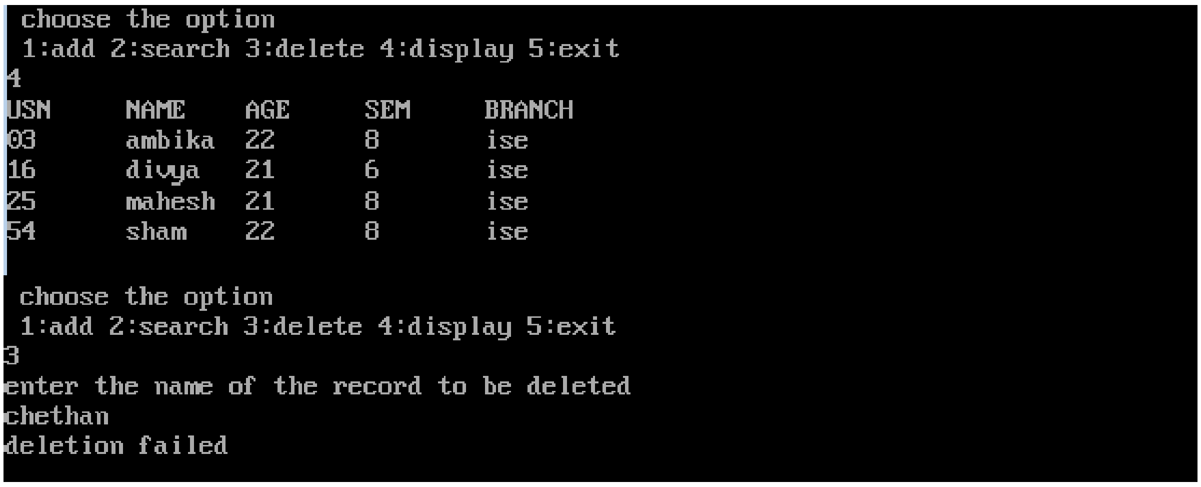
```
 choose the option
 1:add 2:search 3:delete 4:display 5:exit
2
enter the name of the record to be searched
divya
choose the candidates usn
USN:16  NAME:divya
USN:17  NAME:divya
17
USN     NAME    AGE     SEM     BRANCH
17      divya   21      7       cse
choose the candidates usn
USN:16  NAME:divya
USN:17  NAME:divya
16
USN     NAME    AGE     SEM     BRANCH
16      divya   21      6       ise
```



```
 choose the option
 1:add 2:search 3:delete 4:display 5:exit
2
enter the name of the record to be searched
chethan
record search failed
```



```
 choose the option
 1:add 2:search 3:delete 4:display 5:exit
3
enter the name of the record to be deleted
divya
choose the candidates usn
USN:16   NAME:divya
USN:17   NAME:divya
17
record deleted
```



```
 choose the option
 1:add 2:search 3:delete 4:display 5:exit
4
USN     NAME    AGE     SEM     BRANCH
03      ambika  22      8       ise
16      divya   21      6       ise
25      mahesh  21      8       ise
54      sham    22      8       ise

 choose the option
 1:add 2:search 3:delete 4:display 5:exit
3
enter the name of the record to be deleted
chethan
deletion failed
```

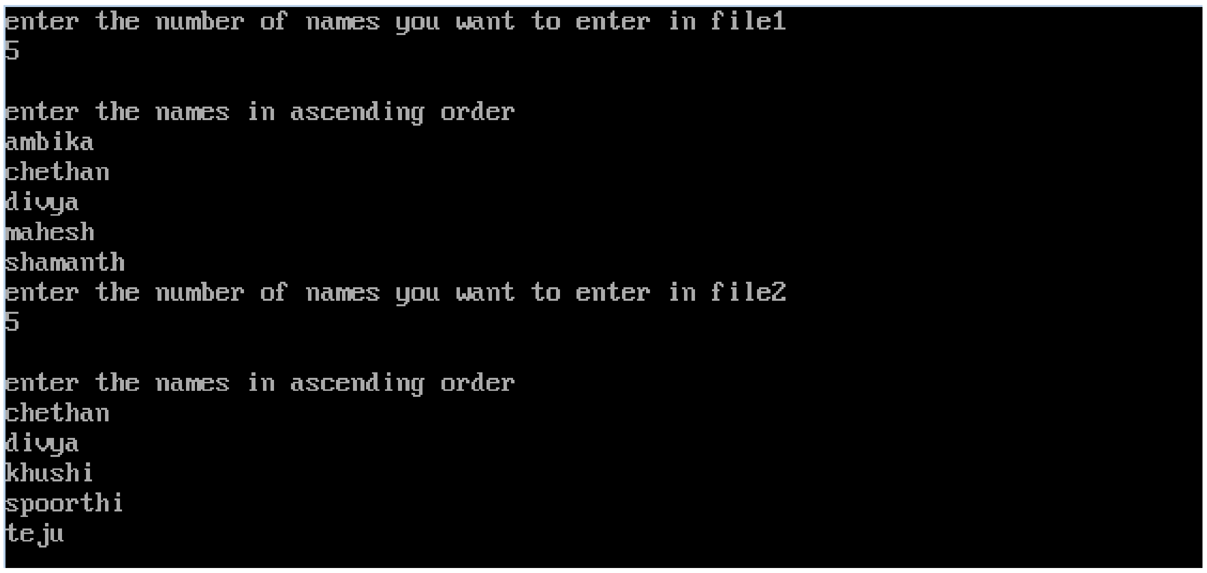
7. Write a C++ program to read two lists of names and then match the names in the two lists using Co-sequential Match based on a single loop. Output the names common to both the lists.

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
#include<fstream.h>
#include<iostream.h>
#include<iomanip.h>

void writeLists()
{
        fstream out1,out2;
        int i;
        int m;
        int n;
        char name[20];

        out1.open("file1.txt",ios::out);
        out2.open("file2.txt",ios::out);

        if( (!out1) || (!out2))
        {
                printf("unable to open one of the list files\n");
                getch();
                exit(0);
        }
        cout<<"enter the number of names you want to enter in file1\n";
        cin>>m;
        cout<<"\nenter the names in ascending order\n";
        for(i=0;i<m;i++)
        {
                cin>>name;
                out1<<name;
                out1<<'\n';
        }
        cout<<"enter the number of names you want to enter in file2\n";
        cin>>n;
        cout<<"\nenter the names in ascending order\n";
        for(i=0;i<n;i++)
        {
                cin>>name;
                out2<<name;
                out2<<'\n';
        }
        out1.close();
        out2.close();
}

void main()
{
```

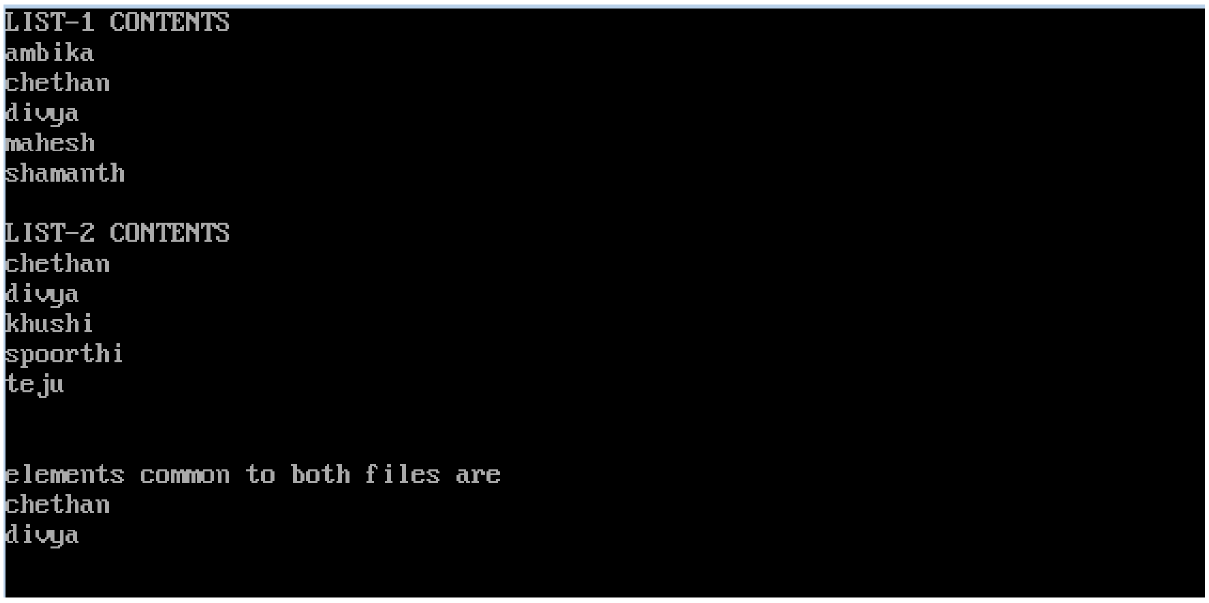
```
char list1[100][20];
char list2[100][20];
int i;
int j;
int m;
int n;
clrscr();
fstream out1,out2,out3;

writeLists();

out1.open("file1.txt",ios::in);
out2.open("file2.txt",ios::in);
out3.open("file3.txt",ios::out);

if( (!out1) || (!out2) || (!out3))
{
        printf("unable to open one of the file");
        getch();
        exit(0);
}

clrscr();
m=0;
n=0;
printf("LIST-1 CONTENTS\n");

while( !out1.eof())
{
        out1.getline(list1[m],20,'\n');
        cout<<list1[m];
        cout<<"\n";
        m++;
}

printf("LIST-2 CONTENTS\n");

while( !out2.eof())
{
        out2.getline(list2[n],20,'\n');
        cout<<list2[n];
        cout<<"\n";
        n++;
}
m--;
n--;
i=0;
j=0;
cout<<"\nelements common to both files are\n";

while(i<m && j<n)
{
        if(strcmp(list1[i],list2[j])==0)
        {
```

```
                        out3<<list1[i];
                        cout<<list1[i]<<"\n";
                        out3<<'\n';
                        i++;
                        j++;
                }
                else if(strcmp(list1[i],list2[j])<0)
                {
                        i++;
                }
                else
                {
                        j++;
                }
        }
        getch();
}
```

OUTPUT:



```
enter the number of names you want to enter in file1
5

enter the names in ascending order
ambika
chethan
divya
mahesh
shamanth
enter the number of names you want to enter in file2
5

enter the names in ascending order
chethan
divya
khushi
spoorthi
teju
```

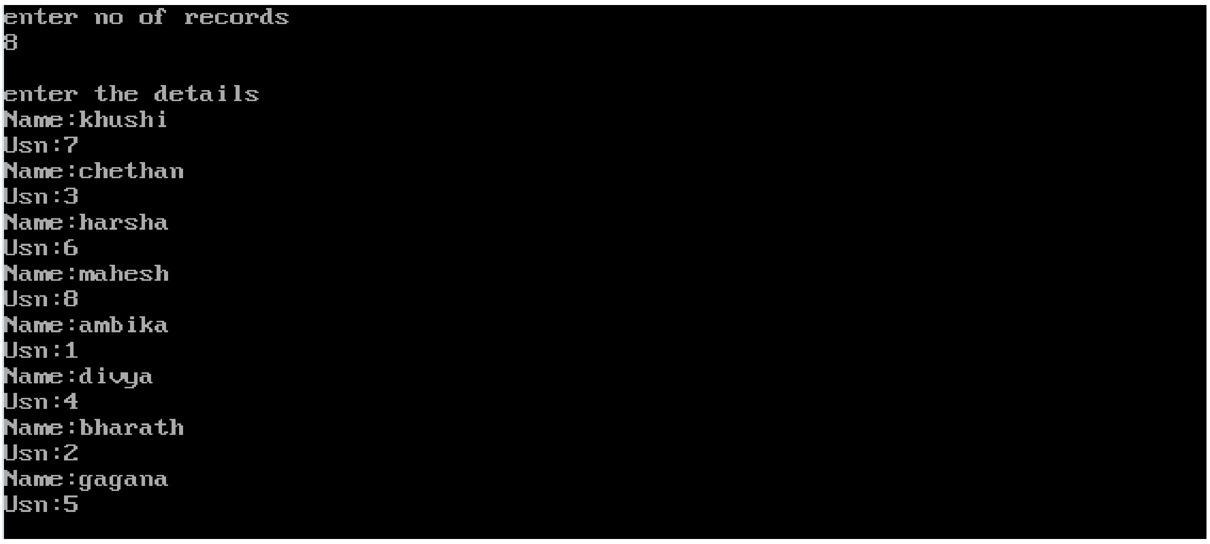


```
LIST-1 CONTENTS
ambika
chethan
divya
mahesh
shamanth

LIST-2 CONTENTS
chethan
divya
khushi
spoorthi
teju


elements common to both files are
chethan
divya
```

8. Write a C++ program to read k Lists of names and merge them using k-way merge algorithm with k = 8.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<fstream.h>
#include<iostream.h>
#include<stdlib.h>

class record
{
        public:
                char name[20];
                char usn[20];
}rec[20];

fstream file[8];
int no;
char fname[8][8]={"1.txt","2.txt","3.txt","4.txt","5.txt","6.txt","7.txt","8.txt"};

void merge_file(char* file1,char* file2,char* filename)
{
        record recd[20];
        int i,k;
        k=0;
        fstream f1,f2;
        f1.open(file1,ios::in);
        f2.open(file2,ios::in);
        while(!f1.eof())
        {
                f1.getline(recd[k].name,20,'|');
                f1.getline(recd[k++].usn,20,'\n');
        }
        while(!f2.eof())
        {
                f2.getline(recd[k].name,20,'|');
                f2.getline(recd[k++].usn,20,'\n');
        }
        int t,y;
        record temp;
        for(t=0;t<k-2;t++)
        for(y=0;y<k-t-2;y++)
        if(strcmp(recd[y].name,recd[y+1].name)>0)
        {
                temp=recd[y];
                recd[y]=recd[y+1];
                recd[y+1]=temp;
        }
        fstream temp1;
        temp1.open(filename,ios::out);
        for(t=1;t<k-1;t++)
        temp1<<recd[t].name<<"|"<<recd[t].usn<<"\n";
        f1.close();
        f2.close();
        temp1.close();
        return;
}

void kwaymerge()
{
        int i,k;
        k=0;
        char filename[7][20]={"11.txt","22.txt","33.txt","44.txt","111.txt","222.txt","1111.txt"};
        for(i=0;i<8;i+=2)
        {
                merge_file(fname[i],fname[i+1],filename[k++]);
        }
        k=4;
        for(i=0;i<4;i+=2)
        {
                merge_file(filename[i],filename[i+1],filename[k++]);
        }
        merge_file(filename[4],filename[5],filename[6]);
        return;
}


int main()
{
        int i;
        clrscr();
        cout<<"enter no of records\n";
        cin>>no;
        cout<<"\nenter the details\n";
        for(i=0;i<8;i++)
        file[i].open(fname[i],ios::out);
        for(i=0;i<no;i++)
        {
                cout<<"Name:";
                cin>>rec[i].name;
                cout<<"Usn:";
                cin>>rec[i].usn;
                file[i%8]<<rec[i].name<<"|"<<rec[i].usn<<"\n";
        }
        for(i=0;i<8;i++)
        file[i].close();
        kwaymerge();
        fstream result;
        result.open("1111.txt",ios::in);
        cout<<"sorted records\n";
        char name[20],usn[20];
        for(i=0;i<no;i++)
        {
                result.getline(name,20,'|');
                result.getline(usn,20,'\n');
                cout<<"\nName:"<<name<<"\nUsn:"<<usn<<"\n";
        }
        getch();
        return 0;
}
```

OUTPUT:



```
enter no of records
8

enter the details
Name:khushi
Usn:7
Name:chethan
Usn:3
Name:harsha
Usn:6
Name:mahesh
Usn:8
Name:ambika
Usn:1
Name:divya
Usn:4
Name:bharath
Usn:2
Name:gagana
Usn:5
```

ADDITIONAL LAB EXPERIMENTS

1. **a)** Modify program 1 to sort the names using selection sort technique

   b) Modify program 1 to sort the reversed names using quick sort technique

2. **a)** Modify program 2 to modify the existing record of the student record by using name as a key and serial number if same name exists for more than one record

   b) Modify program 2 to Search for the particular record of the student using name as a primary key, the output should display more than one record if it exists.

3. **a)** Modify program 3 to use a different delimiter to append at the end the records

   b) Write a program to count number of lines in a file.

4. **a)** Change the name of student in a specified record to a given name

   b) Write a Program to print unique words in a file, include a function that takes a file name as argument and prints all unique words in it.

5. **a)** Implement Indexing using B-Tree Indexes

   b) Write a program to reclaim the free space resulting from the deletion of records using linked lists.

6. **a)** Write a Program to insert student record into the file consisting of Name, USN, Semester, Department, Phone Number and Email Id. Display the content of the file to output terminal, delete a particular record given the line number from a file and update the content of the file.

   b) Write a program to implement File Pointers and Random Access for the content of the given file.

7. **a)** Write a program to read one word at a time from two files (first and second), merge the words and check if they are palindrome. Write the joined palindrome words to third file and non-palindrome words to fourth file. Repeat the operation for remaining words in first and second file.

   b) Write a program to read three lists of names and then match the names in the three lists using Consequential Match based on a single loop. Output the names common to three lists.

8. **a)** Write a program to read list of names from the file, implement k-way splitting technique with specific value for K.

   b) Write a program to append content of three text file to fourth file.

VIVA QUESTIONS

- What is File Structure?
- What is a File?
- What is a field?
- What is a Record?
- What is fixed length record?
- What is RRN?
- What is Variable length record?
- What are the different modes of opening a file?
- What is ifstream()?
- What is ofstream()?
- What is the difference between read() and getline()?
- How to close a file? What happens if a file is not closed?
- What is Hashing? What is its use?
- Explain any one collision resolution technique.
- What is Btree? What is B+tree?
- Differentiate between Logical and Physical file
- What is the use of seekg() and seekp()?
- Explain the different way of write data to a file.
- Explain the different way of write data to a file.
- What is meant by Indexing?
- What is multilevel indexing?
- What is File descriptor?
- What is Fragmentation? What is internal fragmentation?
- What is DMA?
- What is a delimeter?
- Define direct access.

- Define sequential access.
- What is the need of packing the record before writing into the file?
- Explain ios::trunk and ios::nocreate
- What is the use of End-of-file (EOF)?
- What are stdin, stdout and stderr?
- What is Fragmentation?.
- What is data compression?
- What are the properties of B tree?
- How do we delete fixed length records?
- How can we reclaim the deleted space dynamically?
- What are the advantages and disadvantages of indexes that are too large to hold in memory?

- What is an AVL tree?
- H M L B Q S T N A Z P E G C V J K D I U Show B tree creation, insertion, splitting, deletion, merging and redistribution.
- What is memset() ? Explain its parameters.
- What is sprintf() ? Explain its parameters.
- What is the use of tellg() and tellp()?
- What is Boeing tree.