Program 1

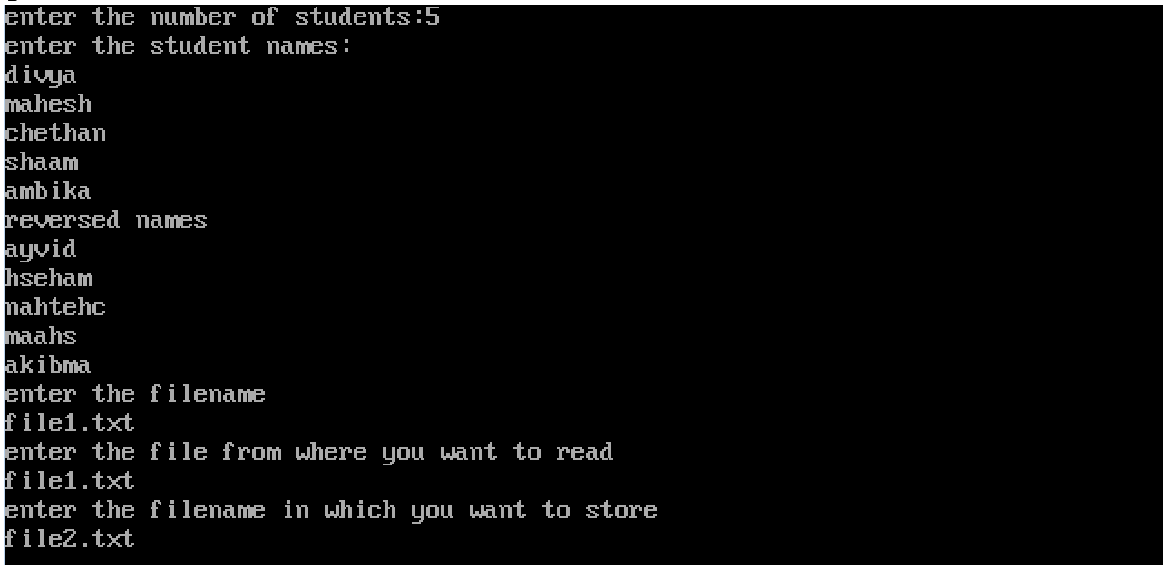
Write a C++ program to read series of names, one per line, from standard input and write these names spelled in reverse order to the standard output using I/O redirection and pipes. Repeat the exercise using an input file specified by the user instead of the standard input and using an output file specified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

Description:
This is 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 pipes.Suppose you would like to change the file listing program so it writes its output to a regular file rather than to stdout. Or suppose you wanted to use the output of the file listing program as input to another program.Because it is common to want to do both of these,operating systems provide convenient shortcuts for switching between standard I/O(stdin and stdout) and regular file I/O.These shortcuts are called I/O redirection and pipes.

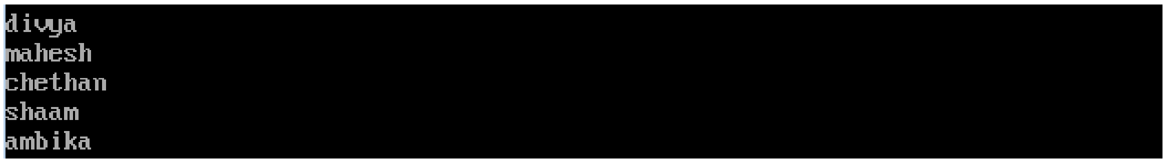
Algorithm:
```
Step 1:start
Step 2:Read n
Step 3:read n student name
Step 4:Read file name
Step 5: if file name existsgoto step 6
        Else goto step 13
Step 6:reverse the student name and store it in file.
Step 7: :read file name from which you want to read as file1
Step 8: if file name existsgoto step 9
        Else goto step 13
Step 9: :read file name where you want to store as file2
Step 10: reverse the student name of file1
Step 11: store reversed names into file2
Step 12:print file2 contents
Step 13:stop.
```

Program:
```
#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
```

Program 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.

Description:

Fixed-Length Records

If we assume that each character occupies one byte, an integer occupies 4 bytes, and a real 8 bytes, our deposit record is 40 bytes long.The simplest approach is to use the first 40 bytes for the first record, the next 40 bytes for the second, and so on.However, there are two problems with this approach.It is difficult to delete a record from this structure.Space occupied must somehow be deleted, or we need to mark deleted records so that they can be ignored.Unless block size is a multiple of 40, some records will cross block boundaries.It would then require two block accesses to read or write such a record. When a record is deleted, we could move all successive records up one which may require moving a lot of records. We could instead move the last record into the ``hole'' created by the deleted record

This changes the order the records are in.It turns out to be undesirable to move records to occupy freed space, as moving requires block accesses.Also, insertions tend to be more frequent than deletions.It is acceptable to leave the space open and wait for a subsequent insertion.This leads to a need for additional structure in our file design.

So one solution is:

At the beginning of a file, allocate some bytes as a **file header**.This header for now need only be used to store the address of the first record whose contents are deleted.This first record can then store the address of the second available record, and so on. To insert a **new** record, we use the record pointed to by the header, and change the header pointer to the **next** available record.If no deleted records exist we add our new record to the end of the file. Fixed-length file insertions and deletions are relatively simple because ``one size fits all''. For variable length, this is not the case.

Algorithm:

STEP 1: Start

STEP 2: Print all options and select one option. If 1 goto step 2, If 2 goto step 3, if 3 goto step 4, if 4 goto step 5. If 0 goto step 6.

STEP 3: (Writing to file) Open a file. Print error message if file doesn't open and goto step 6. Else input student details. Store this record in file with fields delimited by pipe. Remaining bytes in each record is filled with '!' since it is fiexed length record. Close the file and goto step 2.

STEP 3: (Display file) Open the file and print student details using getline. Close the file and goto step 2. If file doesn't open goto step 6.

STEP 4: (Modify record) Open the file. Input usn of record that has to be modified. If record present get details of the record using getline and print the details and enter modifications. Store in the file as in step 3. Close the file and goto step 2. If recoed not present print suitable message and close the file and goto step 2.

STEP 5: (Search record) Open the file. Enter the usn of the recoed that has to be searched. Get details from file using getline. If record is present print record details, close the file and goto step 2. Else print suitable message, close the file and goto step 2.

STEP 6: Stop.

Program:

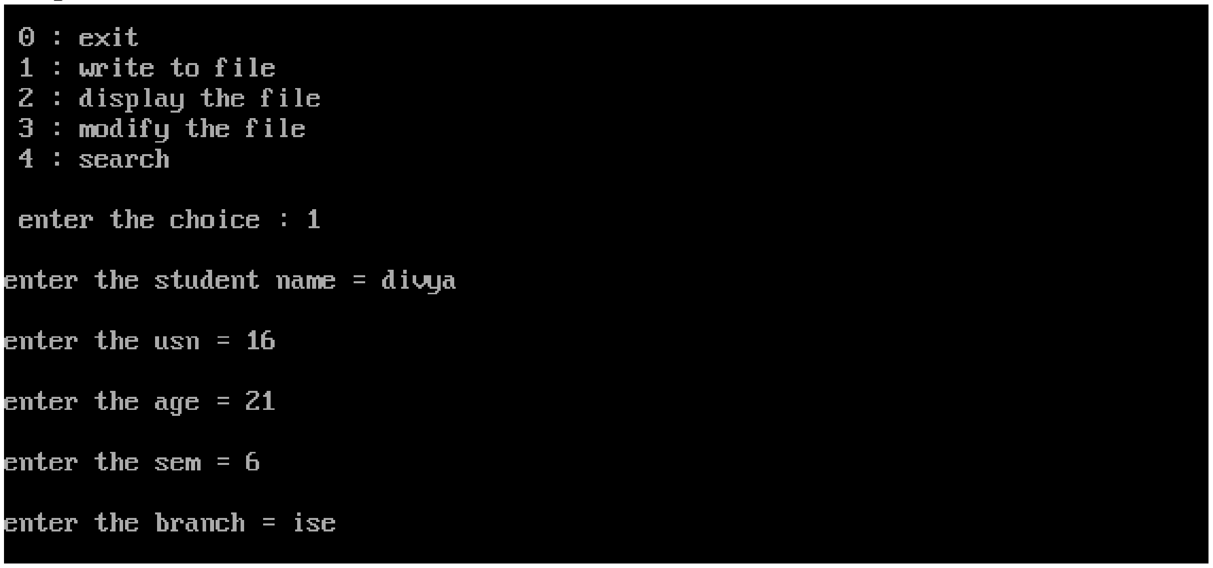
```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<iostream.h>
#include<fstream.h>
#include<string.h>
class student
{
        public: char name[10];
                char usn[10];
                char age[5];
                char sem[5];
                char branch[5];
                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;
        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;
	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)
```

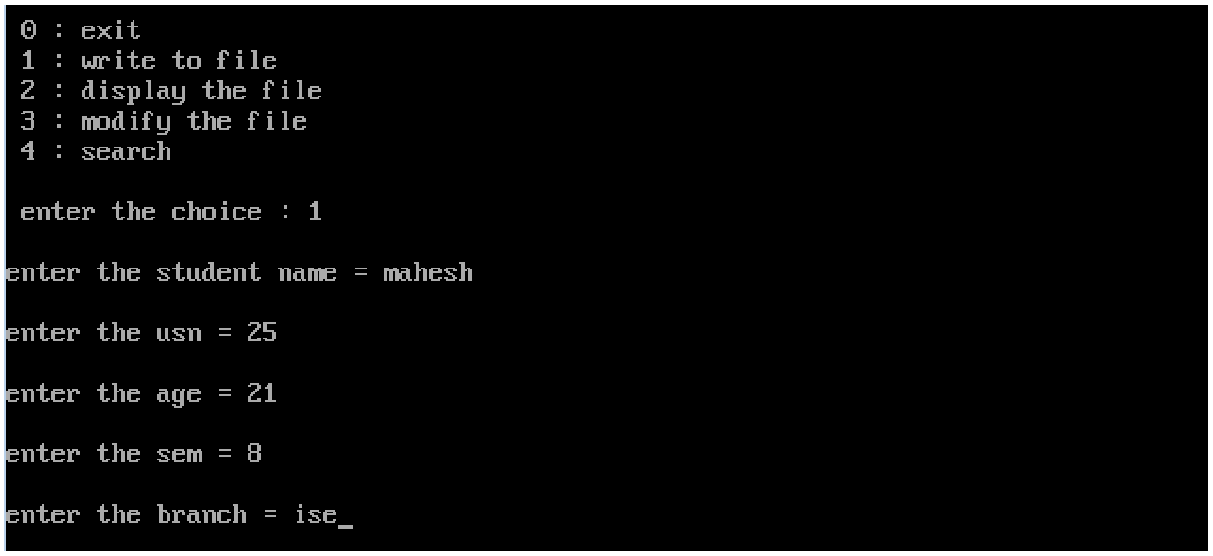
```
        {
                cout<<"\ncannot open the file in read mode";
                getch();
                exit(1);
        }
        i=0;
        cout<<"\n\nNAME\t\tUSN\t\tAGE\t\tSEM\t\tBRANCH\n";
        cout<<"----\t\t---\t\t---\t\t---\t\t------\n";
        while(!file.eof())
        {
                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;
                }
        }
}
```

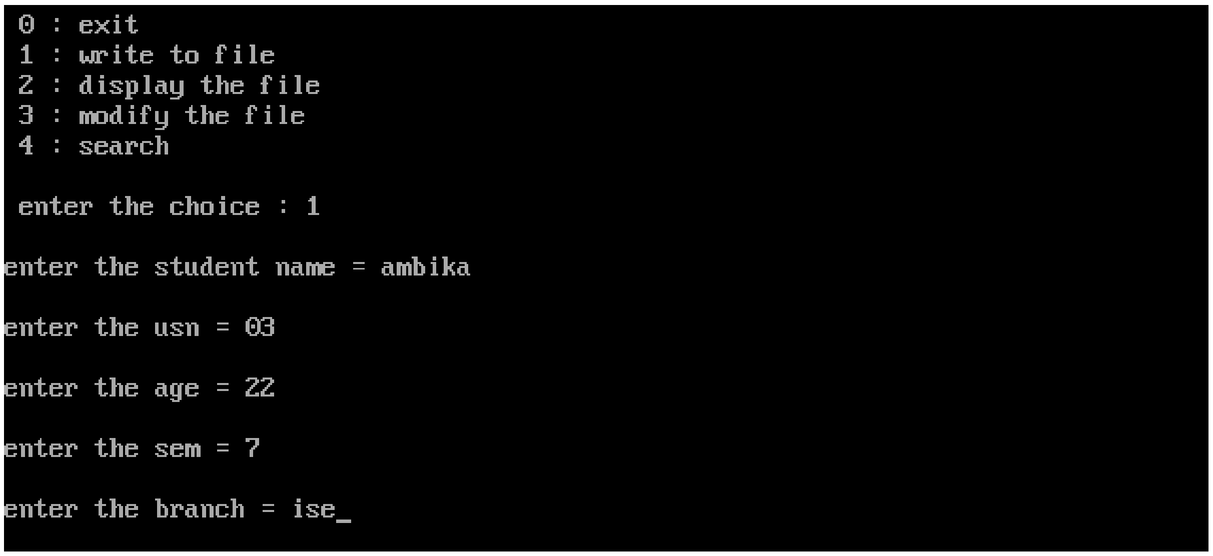
Output:



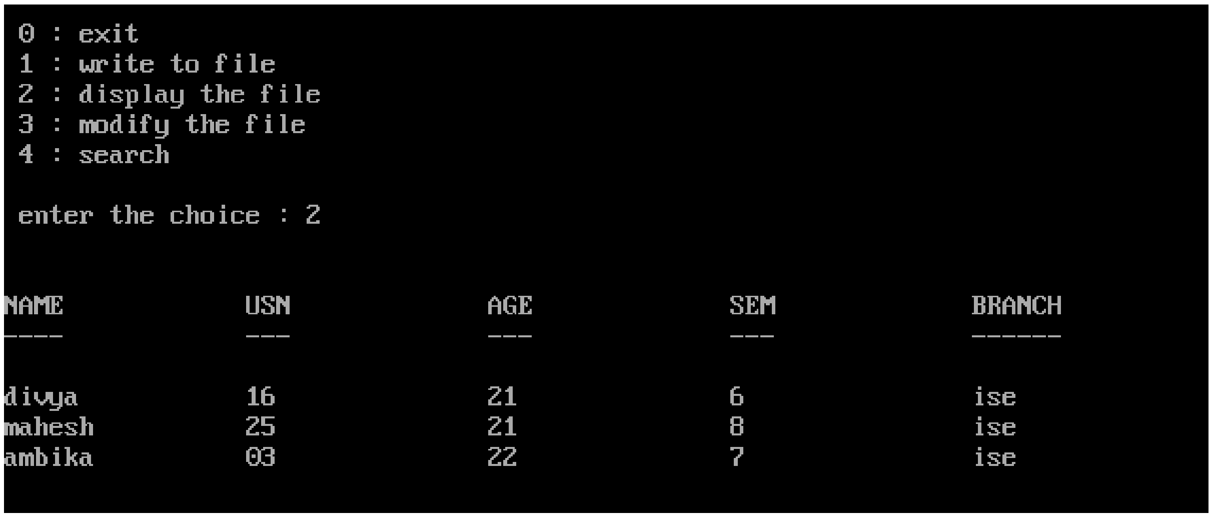
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 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
```



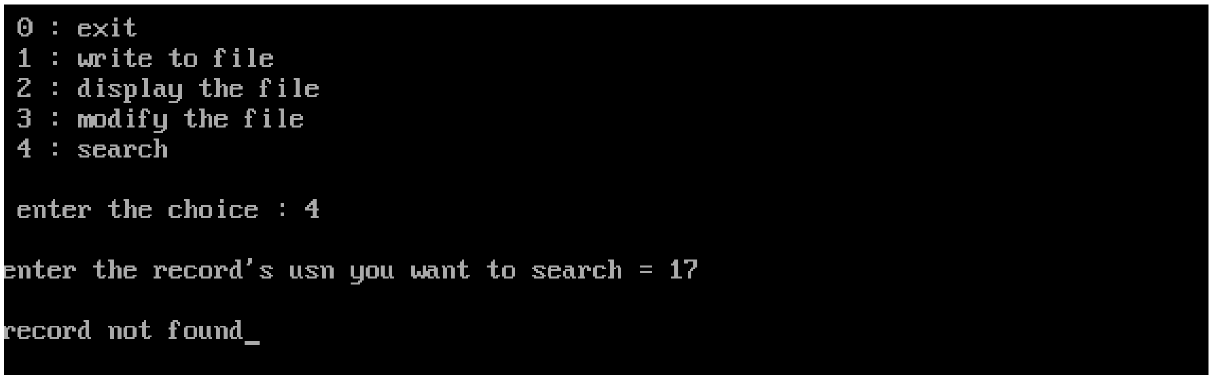
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 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_
```



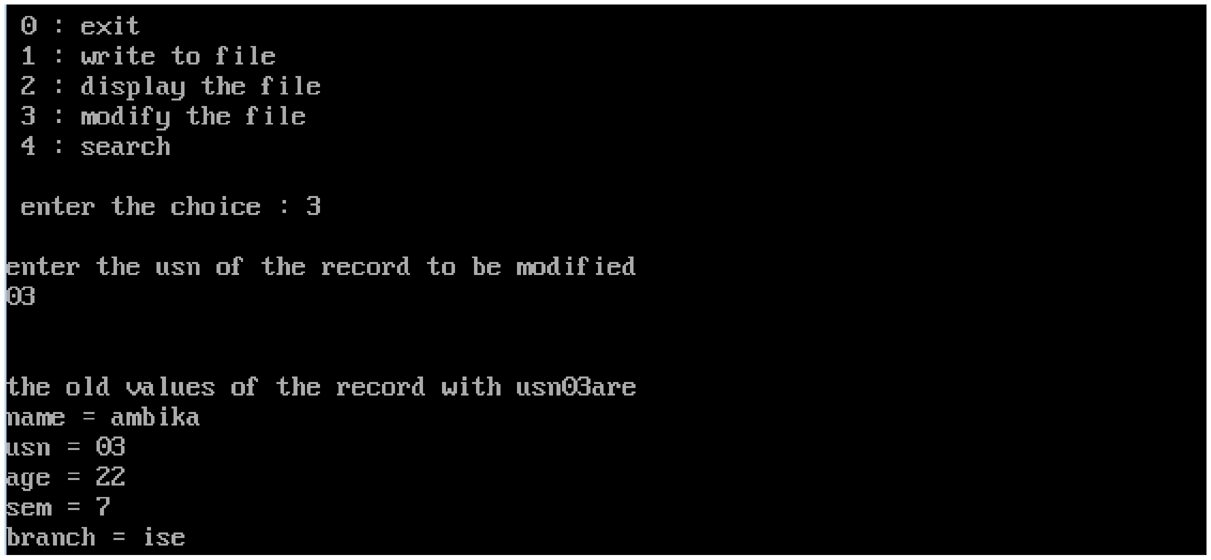
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 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_
```



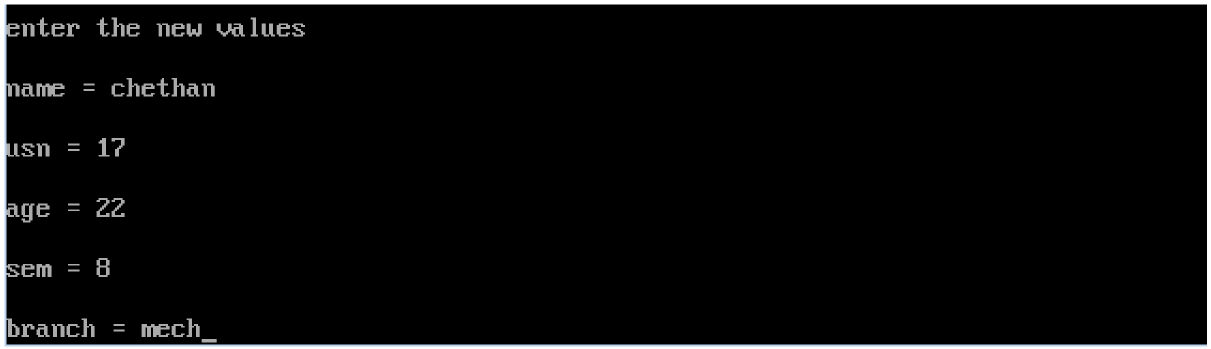
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 2


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

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



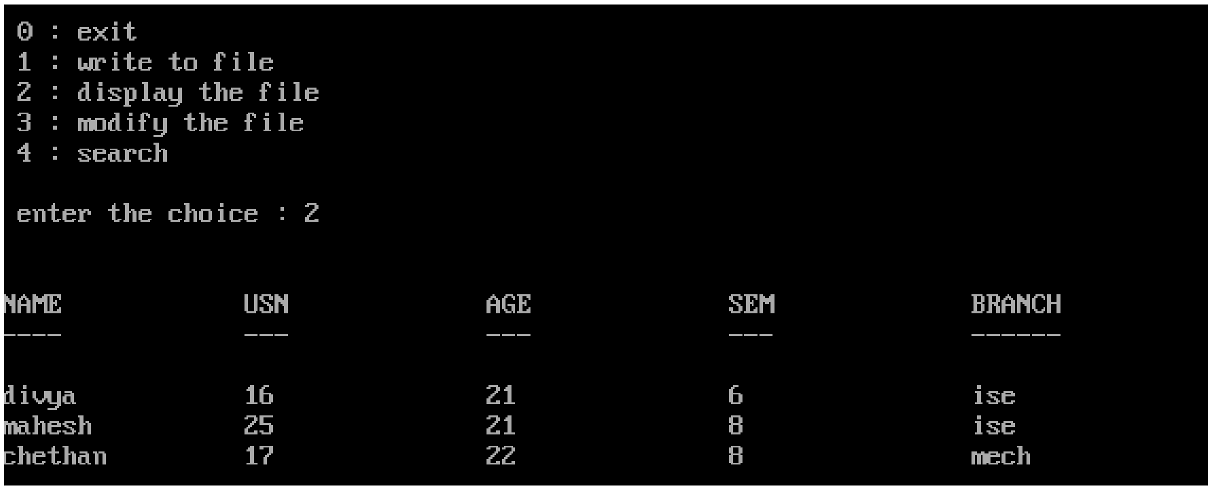
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 4

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

Program 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.

Description

Variable Length Records:
Mechanism for handling an avail list of available space once records are deleted, will apply mechanism to the more complex problem of reusing space from deleted variable-length record.

- A way to link the deleted records together into a list.
- An algorithm for adding newly deleted records to the avail list
- An algorithm for finding and removing records from the avail list when we are ready to use them.

An Avail List of Variable-Length Records
Since we will want to delete whole records and then place records to an avail list, we need a structure in which the record is clearly defined entity. The file structure of Variable-Length Buffer**,** in which we define the length of each record by placing a byte count at the beginning of each record, will serve us well in this regard.
We can handle the contents of a deleted variable-length record just as we did with fixed-length records.
Adding and Removing Records
In variable-length record we have an extra condition that must be met before we can reuse a record: the record must be the right size. We define right size as “big enough”.
We need to search through the avail list for a record slot that is the right size.

Algorithm
STEP 1: Start
STEP 2: Print all options and select one option. If 1 goto step 2, If 2 goto step 3, if 3 goto step 4, if 4 goto step 5. If 0 goto step 6.
STEP 3: (Writing to file) Open a file. Print error message if file doesn't open and goto step 6. Else input student details. Store this record in file with fields delimited by pipe. Close the file and goto step 2.
STEP 4: (Display file) Open the file and print student details using getline. Close the file and goto step 2. If file doesn't open goto step 6.
STEP 5: (Modify record) Open the file. Input usn of record that has to be modified. If record present get details of the record using getline and print the details and enter modifications. Store in the file as in step 3. Close the file and goto step 2. If recoed not present print suitable message and close the file and goto step 2.
STEP 6: (Search record) Open the file. Enter the usn of the record that has to be searched. Get details from file using getline. If record is present print record details, close the file and goto step 2. Else print suitable message, close the file and goto step 2.
STEP 6: Stop.

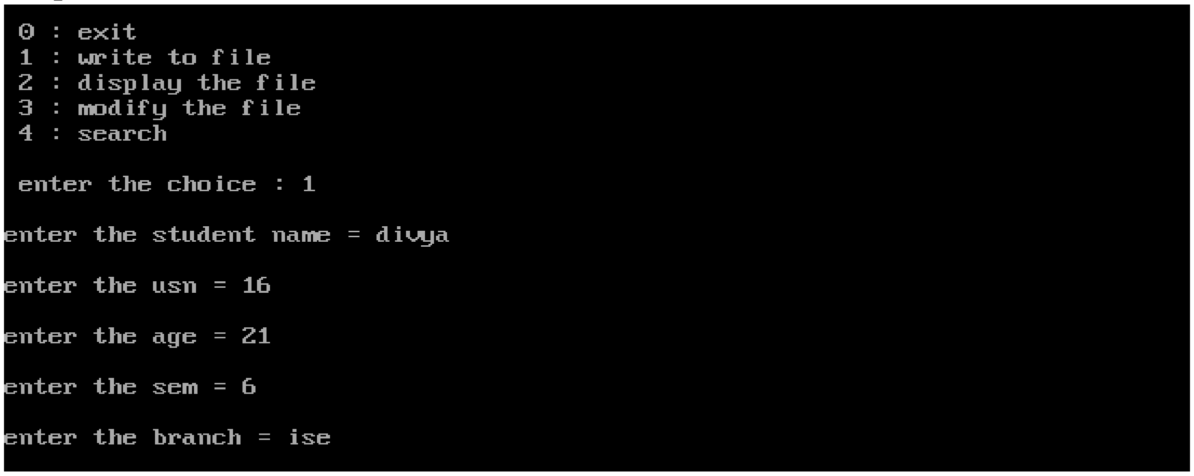
Program:
```
#include<stdio.h>
#include<stdlib.h>
```

```
#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)
        {
```

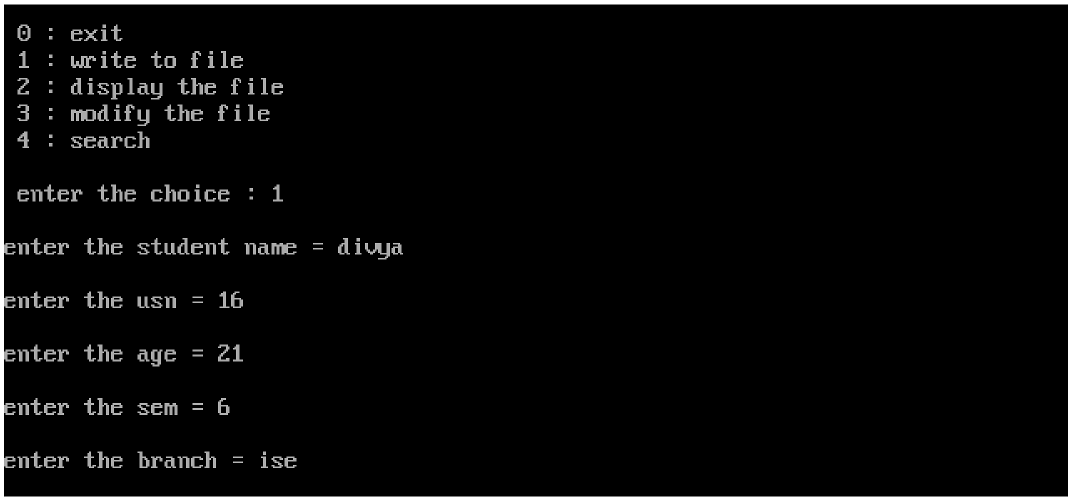
```
		cout<<"\ncannot open the file in read mode";
		getch();
		exit(1);
	}
	i=0;
	printf("\n\nNAME\t\tUSN\t\tAGE\t\tSEM\t\tBRANCH\n");
	while(!file.eof())
	{
		file.getline(s.name,15,'|');
		file.getline(s.usn,15,'|');
		file.getline(s.age,5,'|');
		file.getline(s.sem,5,'|');
		file.getline(s.branch,5,'\n');
		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;
                }
        }
}
```

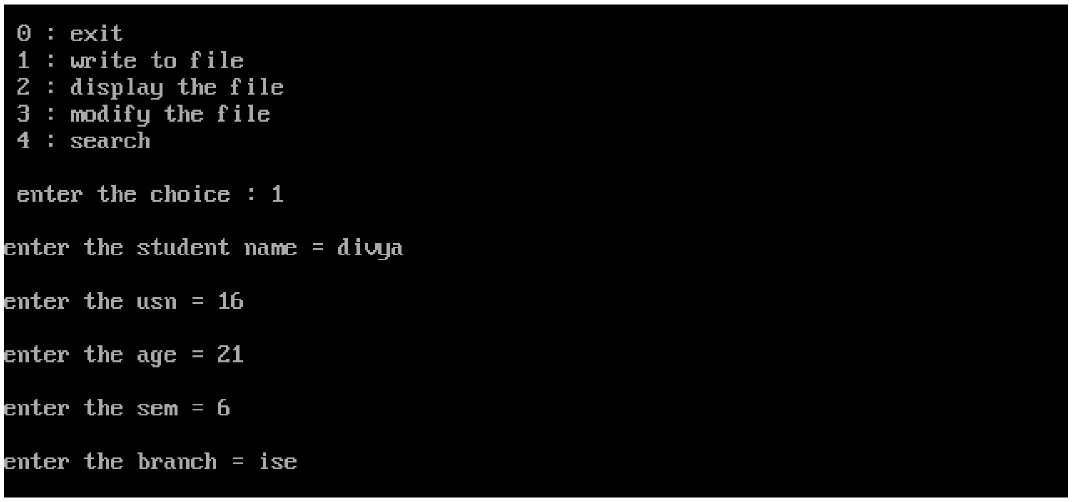
Output:



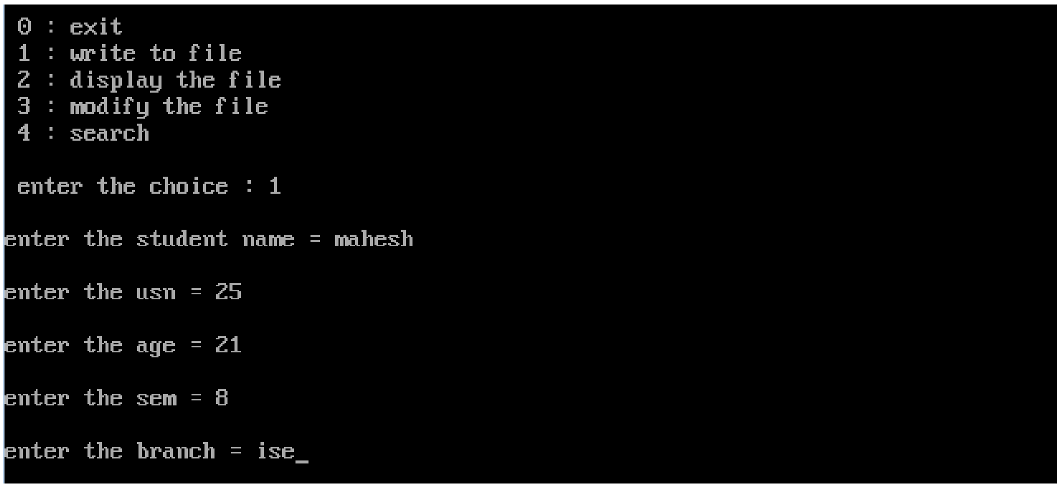
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 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
```



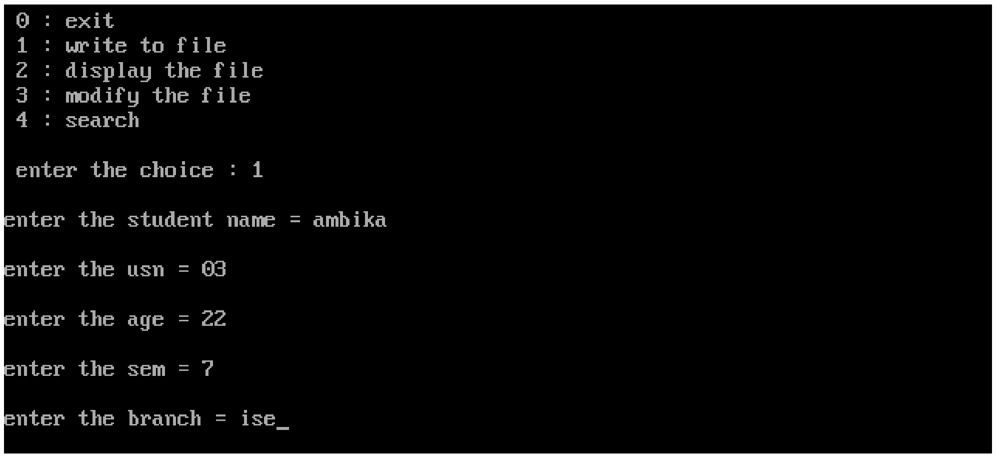
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 1

enter the student name = divya

enter the usn = 16

enter the age = 21

enter the sem = 6

enter the branch = ise
```



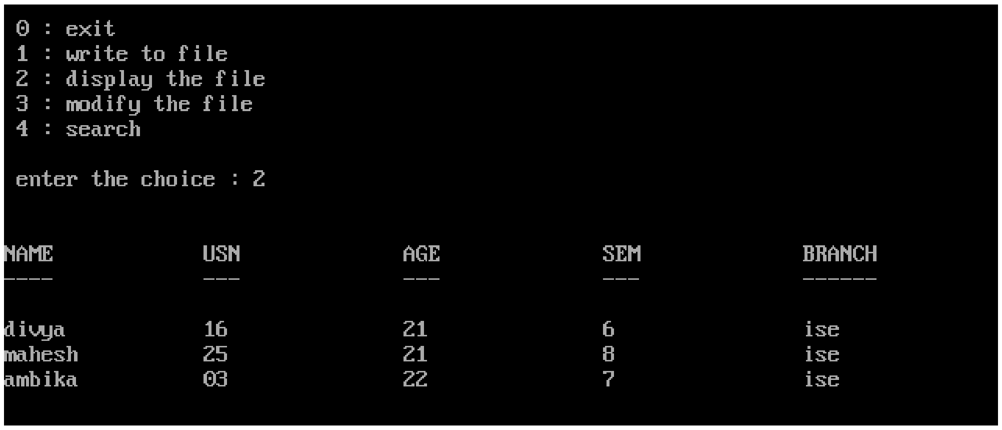
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 1

enter the student name = divya

enter the usn = 16

enter the age = 21

enter the sem = 6

enter the branch = ise
```



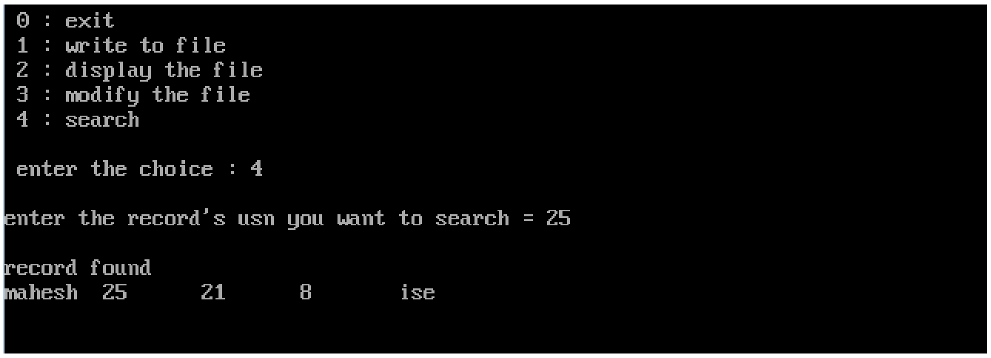
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 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_
```



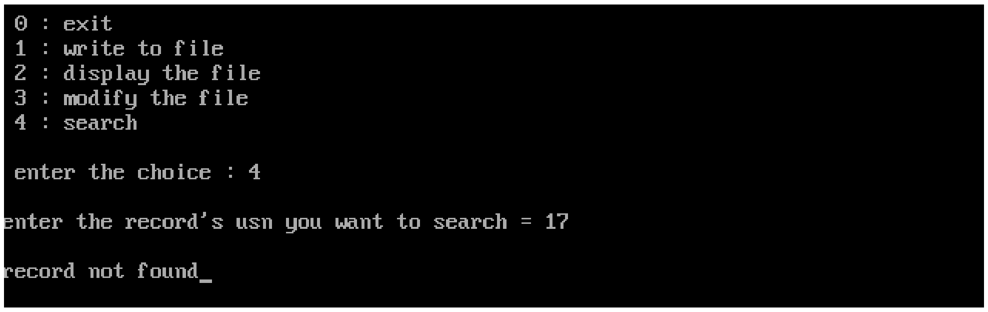
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 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_
```



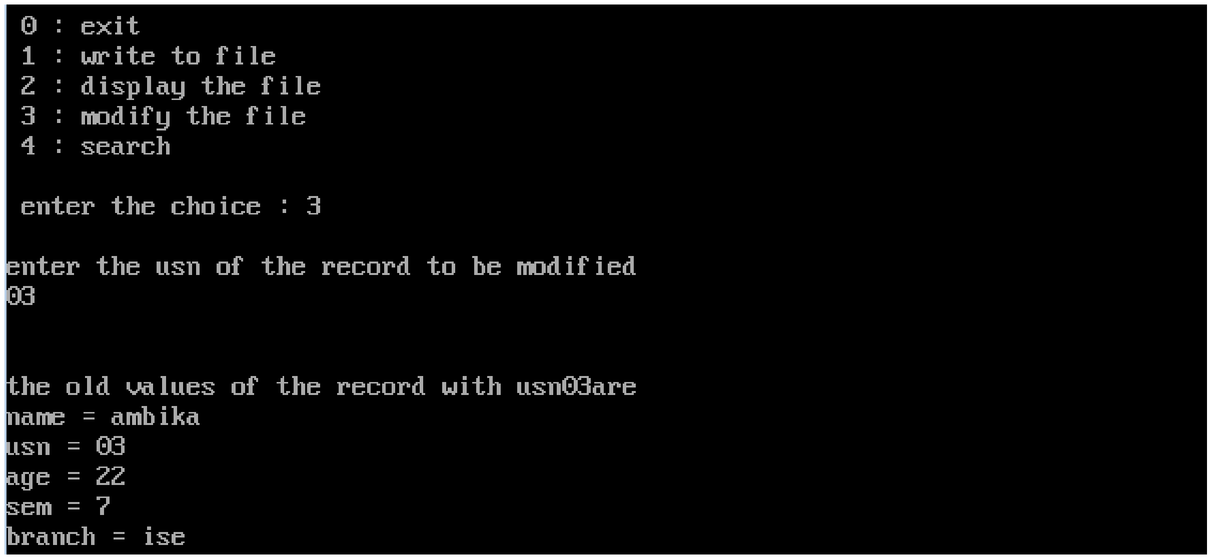
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 2


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

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



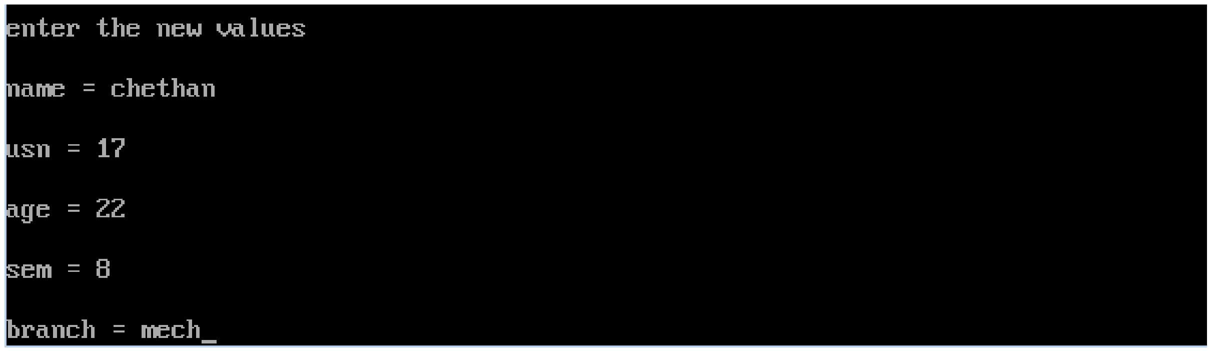
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 enter the choice : 4

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

record found
mahesh  25      21      8       ise
```



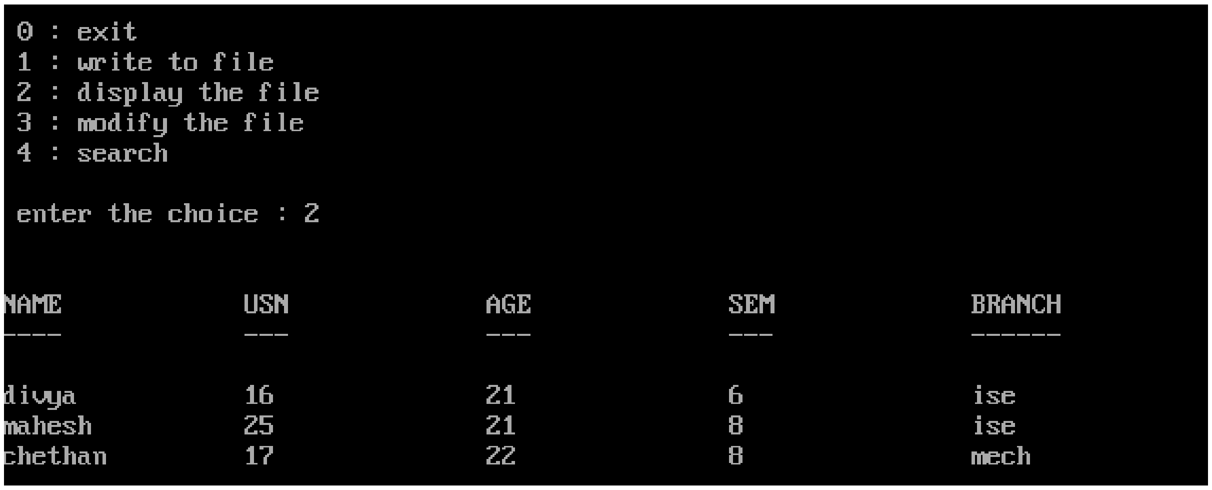
```
 0 : exit
 1 : write to file
 2 : display the file
 3 : modify the file
 4 : search

 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
```

PROGRAM 4

Write a c++ program to write a student object with variable length records using suitable record structure and to read from this file a student record using RRN

Description:

Variable length record:

- Fixed number of fields:

87359|Carroll|Alice in wonderland|38180|Folk|File Structures| ...

- Record beginning with length indicator:

3387359|Carroll|Alice in wonderland|2638180|Folk|File Structures| ..

- Use an index file to keep track of record addresses:
    - The index file keeps the byte offset for each record; this allows us to search the index (which have fixed length records) in order to discover the beginning of the record.
- Placing a delimiter: e.g. end-of-line char

Algorithm:

```
Step1: start
Step2: open a file
Step3: If (!file) go to step4 else go to step 5
Step4: printf("file not found");
Step5: printf("enter the no of records");
Step6: for(i=1to n-1) do
Step7: printf("enter record details");
Step8: go to search
Step9: if(!file) goto step10 else step11
Step10: printf("file not found");
Step11: printf("enter rrn");
Step12: if(rrn) goto step13 else step14
Step13: printf("student details");
Step14: printf("record not found");
Step15: file close
Step16: stop
```

Program:

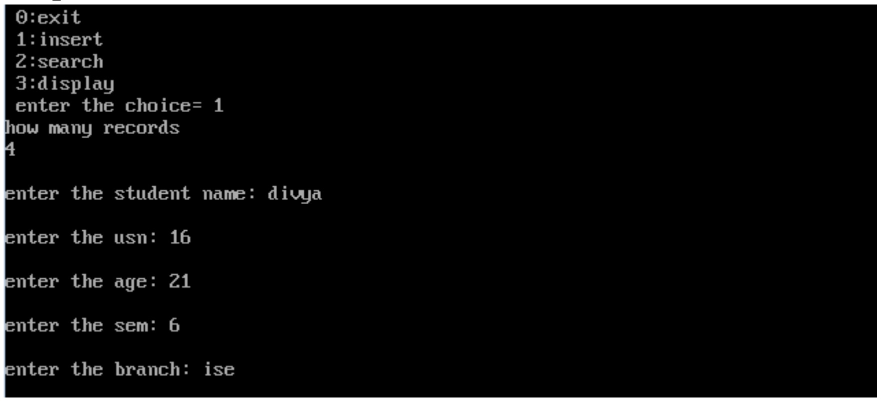
```
#include<stdio.h>
#include<stdlib.h>
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
#include<string.h>
#include<iomanip.h>
class student
{
	public:char name[15],usn[10],age[5],sem[5],branch[15],buffer[100];
```

```
};
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");
```

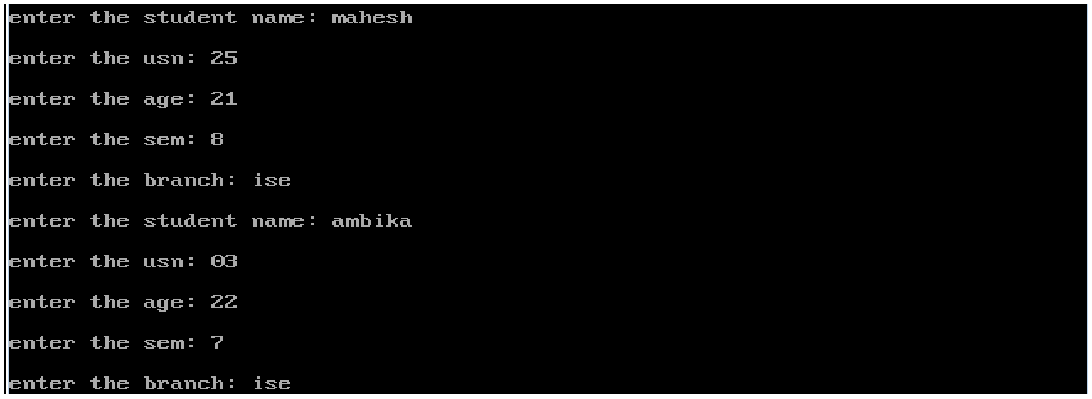
```
	while(!file.eof())
	{
		file.getline(rrn,4,'|');
		file.getline(s.name,15,'|');
		file.getline(s.usn,15,'|');
		file.getline(s.age,5,'|');
		file.getline(s.sem,5,'|');
		file.getline(s.branch,15,'\n');
		printf("\n%s\t%s\t\t%s\t\t%s\t\t%s\t\t%s\n",
		rrn,s.name,s.usn,s.age,s.sem,s.branch);

	}
	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\n 1:insert\n 2:search\n 3:display\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;
                }

        }
}
```

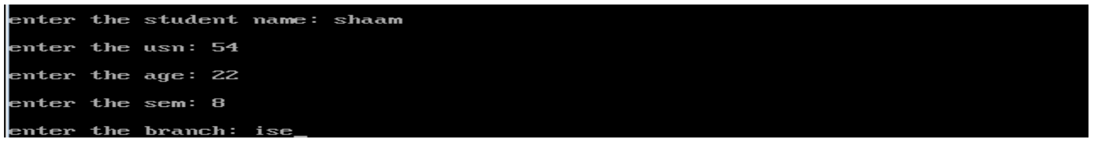
Output:



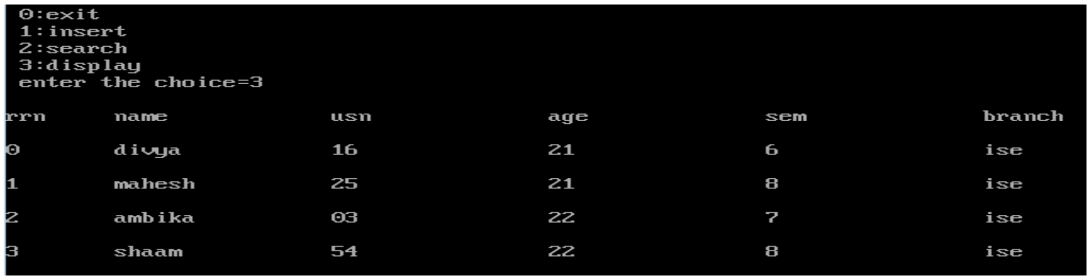
```
 0:exit
 1:insert
 2:search
 3:display
 enter the choice= 1
how many records
4

enter the student name: divya

enter the usn: 16

enter the age: 21

enter the sem: 6

enter the branch: ise
```



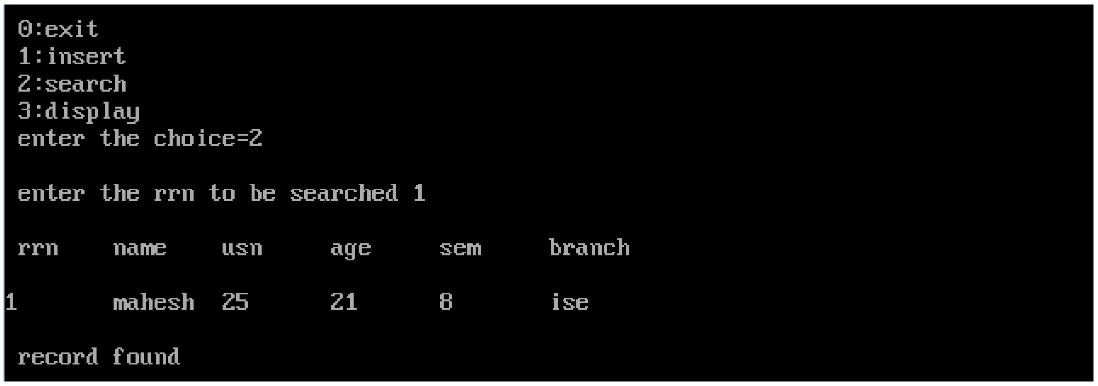
```
enter the student name: mahesh

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
```



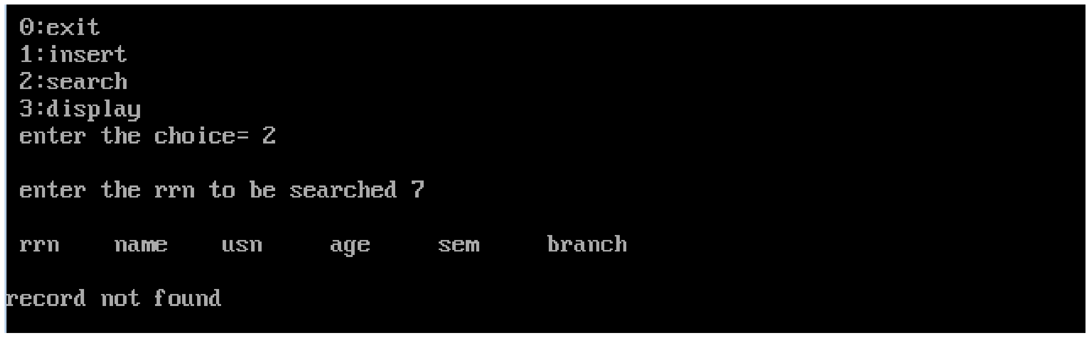
```
enter the student name: shaam
enter the usn: 54
enter the age: 22
enter the sem: 8
enter the branch: ise_
```



```
 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
```

Program 5

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

Description:

Say that a file with variable length records resides on the hard disk. This file contains student records and the structure of each record in this file is as described in Program 4. To search this file for a particular USN, we need to read in record by record and compare the USN in each record with the search USN, i.e., we need to do linear search on the records. But if binary search is to be performed, the records must be sorted. Sorting the records in the file every time a new record is inserted is too expensive. To delete a record, we would have to first search for it using linear search, then delete the record.

The solution is to maintain an USN-based index of the file in main memory, where sorting, and hence searching, is faster. Such an index is referred to as the primary index, because the USN is the primary key of the student database. Once the index is sorted on USN, binary search can be used for searching. When new records are inserted into the file, an appropriate entry is also added to the index in a way so as to maintain the sorted order of the index. When a record is deleted from the file, the corresponding entry in the index is also deleted and all the index entries below the deleted entry are pushed up by one (still maintaining sorted order). We implement this technique in this program.

We first create a structure called list, which contains a character field of 15 spaces to store a primary key (USN, in our case) and an integer field to store the byte offset of each record in the file. When we access an index entry, we get the byte offset of the record to which this index is pointing to, and we use seekg() to jump to the record directly without walking through the file in a serial manner. We create 100 elements of this list here, but if more than 100 records are to be indexed, then this number has to be increased. To know how many index elements are present, we use a global variable called tpiiu and initialize it to 0 as the index is empty to start with.

Algorithm:

```
Step 1 :  start
Step 2: create two files for record and index file
        index file contain primary key,secondary
        key,rrn and record file contain all attributes.
step 3 :search the file according to usn
Step4: to search the record required we will
       search the index file based on secondary key.if
       only one record found  goto step 6 else goto
Step 5:search based on the primary key in index file.
step 6:compare rrn of record file and index  file.
Step 7:display the content of record and index file.Goto step 10.
Step 8:to delete record first perform search
Step 9:match the rrn of record file and selected
Record's  rrn.if match found delete the record
Else print error message.goto step 7.
```

step 10:stop

Program:

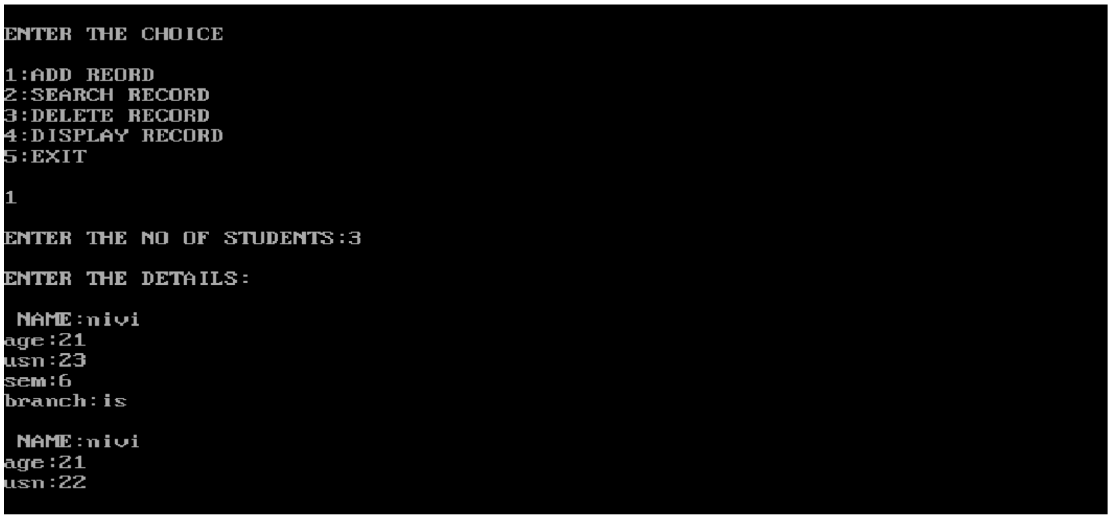
```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>
#include<fstream.h>
#include<iostream.h>
#include<iomanip.h>
class record
{
	public:		char age[5];
		char usn[20];
		char name[20];
		char sem[2];
		char branch[5];
}rec[20];
char st_no[5];
int no;
void retrieve_details()
{
	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";
```

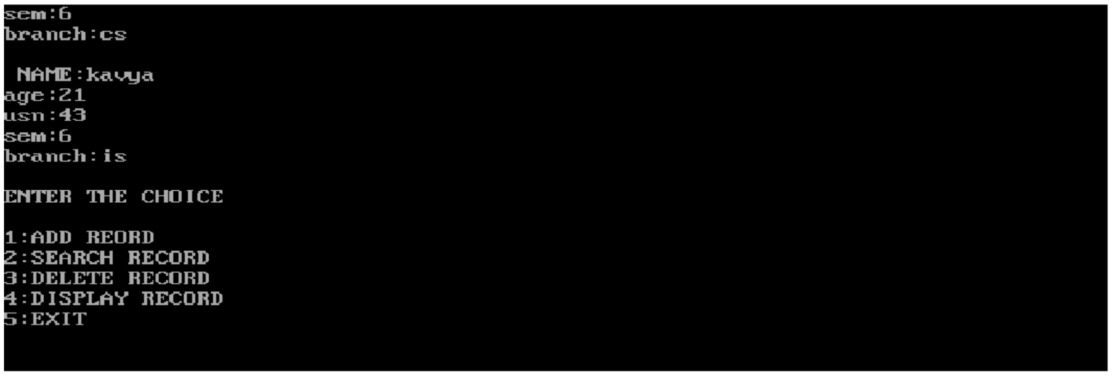
```
cout<<"4 : display record\n";
cout<<"5 : exit\n\n";
cin>>choice;
switch(choice)
{
        case 1: cout<<"\nenter the no. of students : ";
                cin>>no;
                cout<<"\nenter the details :\n";
                for(i=0;i<no;i++)
                {
                        cout<<"\nname :";
                        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;
     }
  }
}
```

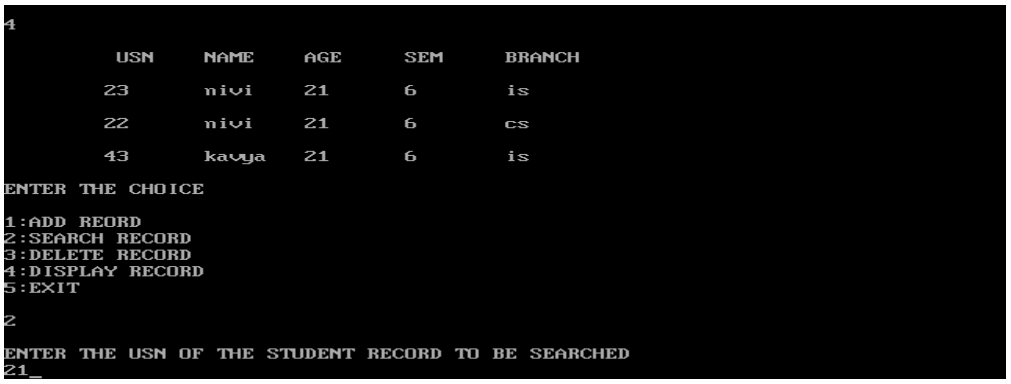
Output:



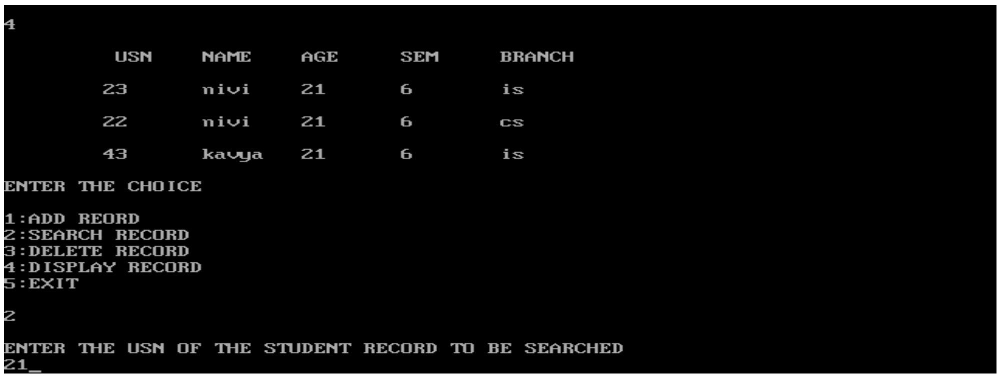
```
ENTER THE CHOICE

1:ADD REORD
2:SEARCH RECORD
3:DELETE RECORD
4:DISPLAY RECORD
5:EXIT

1

ENTER THE NO OF STUDENTS:3

ENTER THE DETAILS:

 NAME:nivi
age:21
usn:23
sem:6
branch:is

 NAME:nivi
age:21
usn:22
```



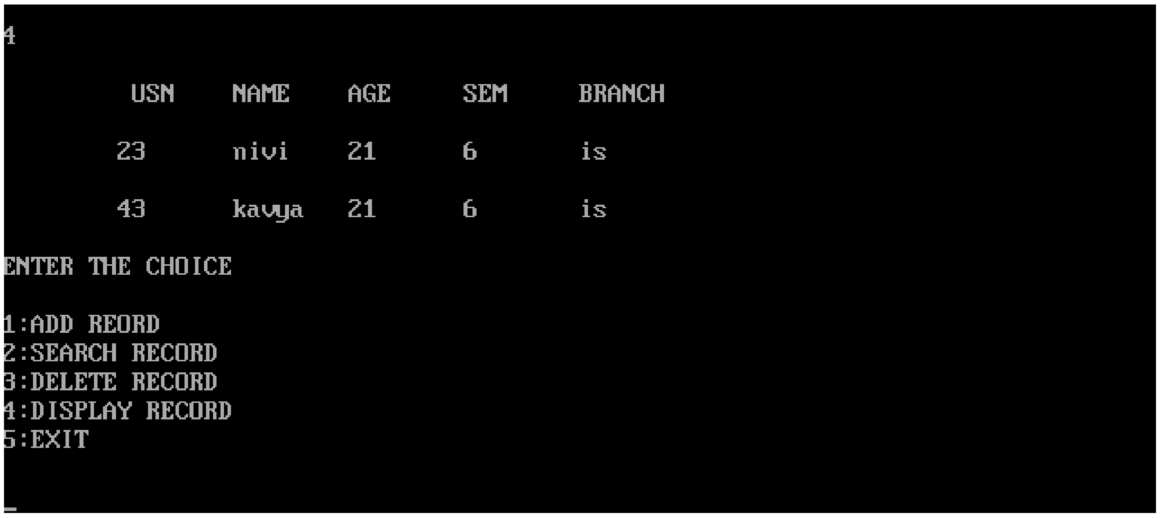
```
sem:6
branch:cs

 NAME:kavya
age:21
usn:43
sem:6
branch:is

ENTER THE CHOICE

1:ADD REORD
2:SEARCH RECORD
3:DELETE RECORD
4:DISPLAY RECORD
5:EXIT
```



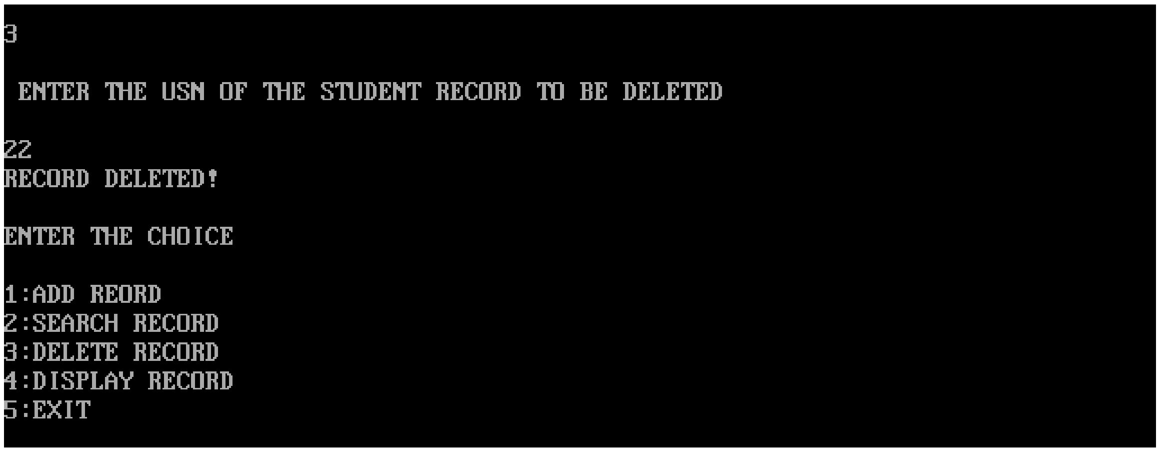
```
4
         USN     NAME     AGE      SEM      BRANCH

        23       nivi     21       6        is

        22       nivi     21       6        cs

        43       kavya    21       6        is

ENTER THE CHOICE

1:ADD REORD
2:SEARCH RECORD
3:DELETE RECORD
4:DISPLAY RECORD
5:EXIT

2

ENTER THE USN OF THE STUDENT RECORD TO BE SEARCHED
21_
```



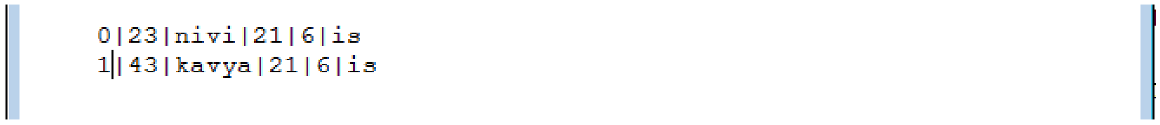
```
4
         USN     NAME     AGE      SEM      BRANCH

        23       nivi     21       6        is

        22       nivi     21       6        cs

        43       kavya    21       6        is

ENTER THE CHOICE

1:ADD REORD
2:SEARCH RECORD
3:DELETE RECORD
4:DISPLAY RECORD
5:EXIT

2

ENTER THE USN OF THE STUDENT RECORD TO BE SEARCHED
21_
```



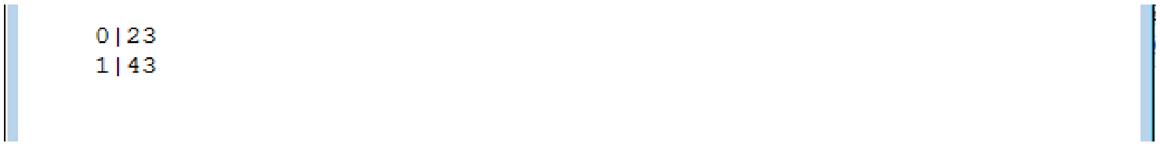
```
4

         USN     NAME     AGE      SEM      BRANCH

        23       nivi     21       6        is

        43       kavya    21       6        is

ENTER THE CHOICE

1:ADD REORD
2:SEARCH RECORD
3:DELETE RECORD
4:DISPLAY RECORD
5:EXIT

_
```



```
3

 ENTER THE USN OF THE STUDENT RECORD TO BE DELETED

22
RECORD DELETED!

ENTER THE CHOICE

1:ADD REORD
2:SEARCH RECORD
3:DELETE RECORD
4:DISPLAY RECORD
5:EXIT
```



```
0|23|nivi|21|6|is
1|43|kavya|21|6|is
```



```
0|23
1|43
```

Program 6

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

Description:

This program explains about the implementation of secondary key on index file structure.Any field in a record may be a secondary key. The problem with secondary keys is that they are not unique and are therefore likely to return more than one record for a particular value of the key. Some fields have a large enough range of values that a search for a specific value will produce only a few records; other fields have a very limited range of values and a search for a specific value will return a large proportion of the file. An example of the latter would would be a search in student records for students classified as freshmen.The answer in most cases lies is preprocessing to create some kind of aid in the search. The primary indexes we have seen in extendible hashing and indexed sequential access systems are examples of preprocessing to achieve efficiency of access later. Primary indexes offer acceptable returns on the processing investment and space costs because the maintenance cost is low and the usage is high. Secondary indexes are a different story; because secondary keys are not necessarily based on unchanging data, the index maintenance requirement may be considerable. Further, access on every field in a record is unlikely. Because of the maintenance requirement and lack of use, indexing on all fields is probably impractical and wasteful. Therefore, it is important that the file designer investigate to determine which fields are likely candidates for frequent secondary key access in order to justify the processing and space costs and to determine the appropriate maintenance routines.

Algorithm:

```
Step 1 :  start
Step 2: create two files for record and index file
        index file contain primary key,secondary
        key,rrn and record file contain all attributes.
step 3 :sort the file according to names
Step4: to search the record required we will
       search the index file based on secondary key.if
       only one record found  goto step 6 else goto
Step 5:search based on the primary key in index file.
step 6:compare rrn of record file and index  file.
Step 7:display the content of record and index file.
Goto step 10.
Step 8:to delete record first perform search
Step 9:match the rrn of record file and selected
Record's  rrn.if match found delete the record
Else print error message.goto step 7.
step 10:stop
```

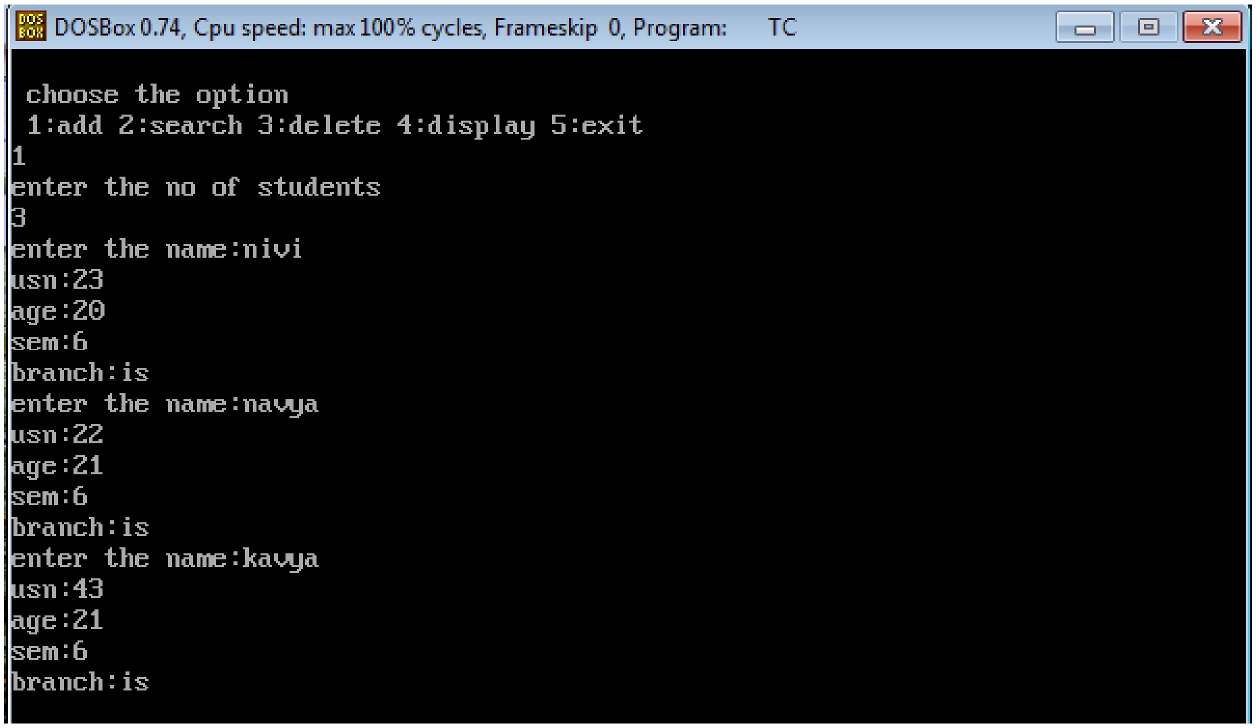
Program:

```
#include<stdio.h>
#include<conio.h>
```

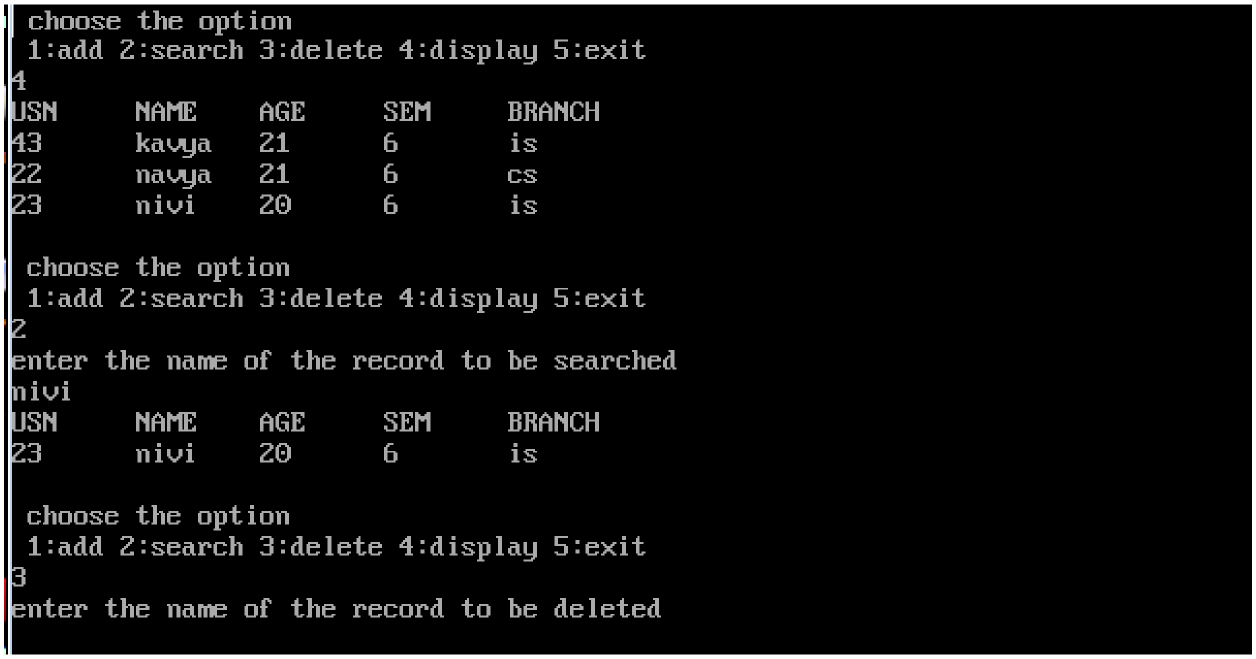
```
#include<string.h>
#include<fstream.h>
#include<iostream.h>
#include<stdlib.h>
class record
{
        public:
                char age[5],sem[5],usn[20],name[20],branch[5];
}rec[20],found[20];
int no;
char st_no[5],rt_name[20];
void sortrecord()
{
        int i,j;
        record temp;
        for(i=0;i<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;
                    }
          }
}
```

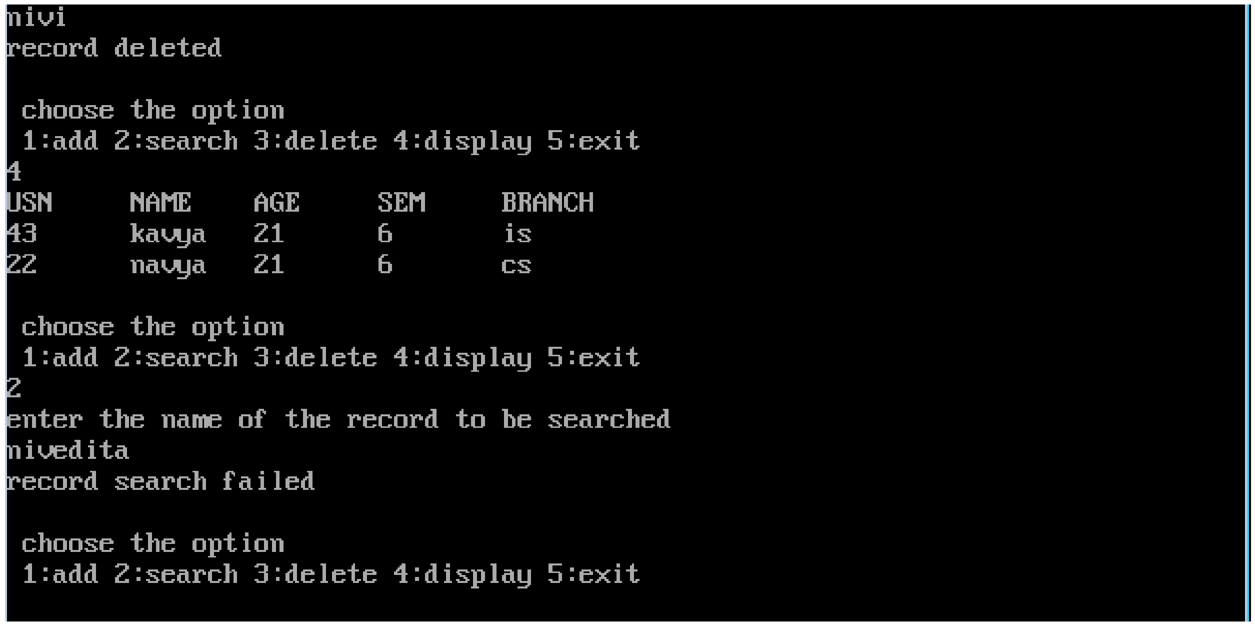
OUTPUT:



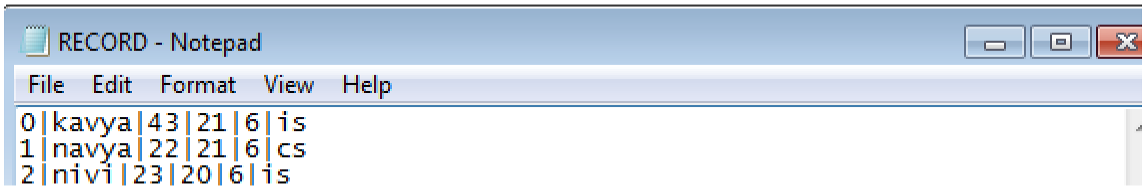
```
 choose the option
 1:add 2:search 3:delete 4:display 5:exit
1
enter the no of students
3
enter the name:nivi
usn:23
age:20
sem:6
branch:is
enter the name:navya
usn:22
age:21
sem:6
branch:is
enter the name:kavya
usn:43
age:21
sem:6
branch:is
```



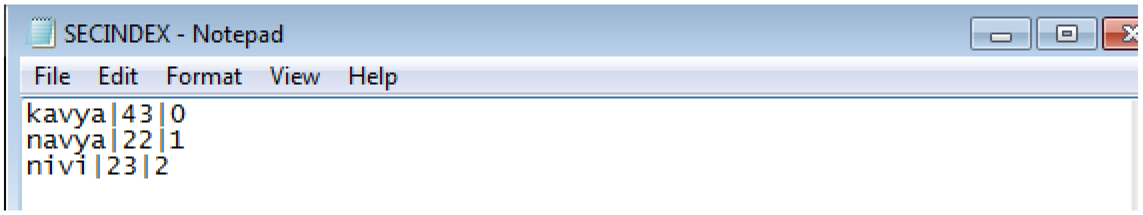
```
 choose the option
 1:add 2:search 3:delete 4:display 5:exit
4
USN     NAME    AGE     SEM     BRANCH
43      kavya   21      6       is
22      navya   21      6       cs
23      nivi    20      6       is

 choose the option
 1:add 2:search 3:delete 4:display 5:exit
2
enter the name of the record to be searched
nivi
USN     NAME    AGE     SEM     BRANCH
23      nivi    20      6       is

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



```
nivi
record deleted

 choose the option
 1:add 2:search 3:delete 4:display 5:exit
4
USN     NAME    AGE     SEM     BRANCH
43      kavya   21      6       is
22      navya   21      6       cs

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

 choose the option
 1:add 2:search 3:delete 4:display 5:exit
```



```
0|kavya|43|21|6|is
1|navya|22|21|6|cs
2|nivi|23|20|6|is
```



```
kavya|43|0
navya|22|1
nivi|23|2
```

Program 7

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.

Description:

This program is based on the theory of **consequential operations**

Consequential operations:

Consequential operation is nothing but coordinated processing of two or more sequential lists to produce a single list .

Kinds of operations:

- merging, or union
- matching, or intersection
- combination of above

This program deals with intersection operation:

Matching Names in Two Lists:

In comparing two names

- if Item(1) is less than Item(2), read the next from List 1
- if Item(1) is greater than Item(2), read the next name from List 2
- if the names are the same, output the name and read the next names from the two lists

Algorithm:

```
STEP 1:  Start
STEP 2: Create two files as file1&file2.
STEP 3:Enter the number & names of files you want to enter in file1 in ascending
order go to step 4
STEP 4: Print the names of file1 go to step 5
STEP 5:Enter the number & name of files you want to enter in file2 in ascending
order. Goto step 6
STEP 6: Print the names of file2 go to step 7
STEP 7: Close the files goto step 8.
STEP 8: Create another file as file3 to store the output.goto step9
STEP 9: Open the 3 files. If we are unable to open the files then print an error message
        else goto step10.
STEP 10:Print all the contents of list1 by extracting the contents of list1 until new
line is encountered.Goto step11.
STEP 11: Follow the step10 procedure for printing list2 contents.
STEP 12: Initialize i&j to zero goto step13.
STEP 13: Print the elements common to both files. Compare list1&list2 contents.
If they are equal to zero ,put the contents of list1 to 3rd file &goto step 16 else goto step 14.
STEP 14:If list1&list2 contents are less than zero incerment i  then goto step 16.else goto step 15
STEP 15: Increment j then goto step 16
STEP 16: Stop.
```

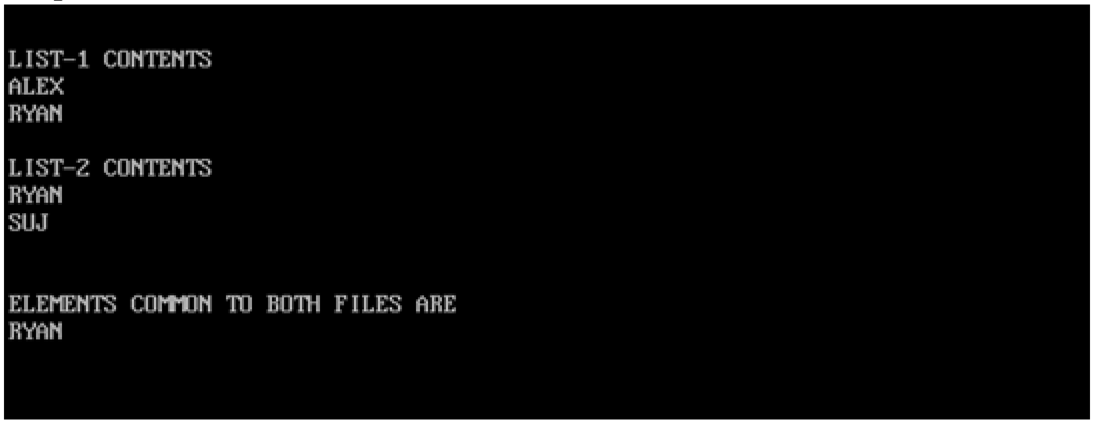
Program:

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

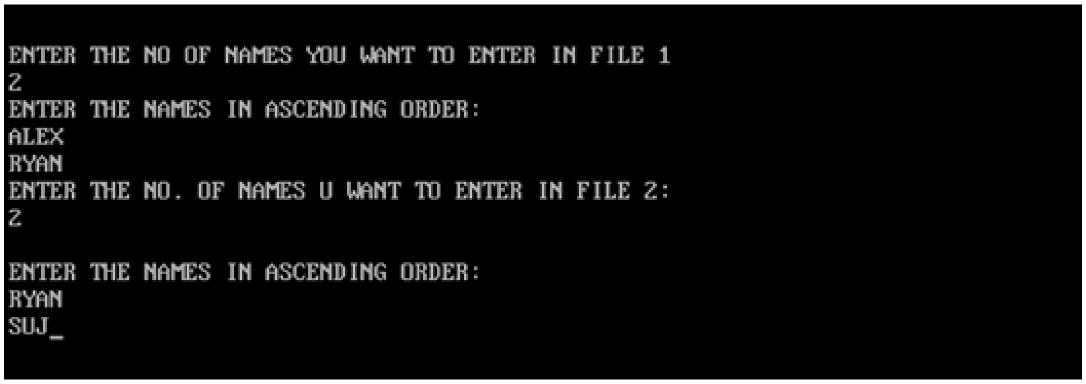
```
#include<string.h>
#include<fstream.h>
#include<iostream.h>
#include<iomanip.h>
void writeLists()
{
        fstream out1,out2;
        int i;
        int m;
        int n;
        char name[20];
        out1.open("file1.txt",ios::out);
        out2.open("file2.txt",ios::out);
        if( (!out1) || (!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:



```
LIST-1 CONTENTS
ALEX
RYAN

LIST-2 CONTENTS
RYAN
SUJ


ELEMENTS COMMON TO BOTH FILES ARE
RYAN
```



```
ENTER THE NO OF NAMES YOU WANT TO ENTER IN FILE 1
2
ENTER THE NAMES IN ASCENDING ORDER:
ALEX
RYAN
ENTER THE NO. OF NAMES U WANT TO ENTER IN FILE 2:
2

ENTER THE NAMES IN ASCENDING ORDER:
RYAN
SUJ_
```

Program 8

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

Description:

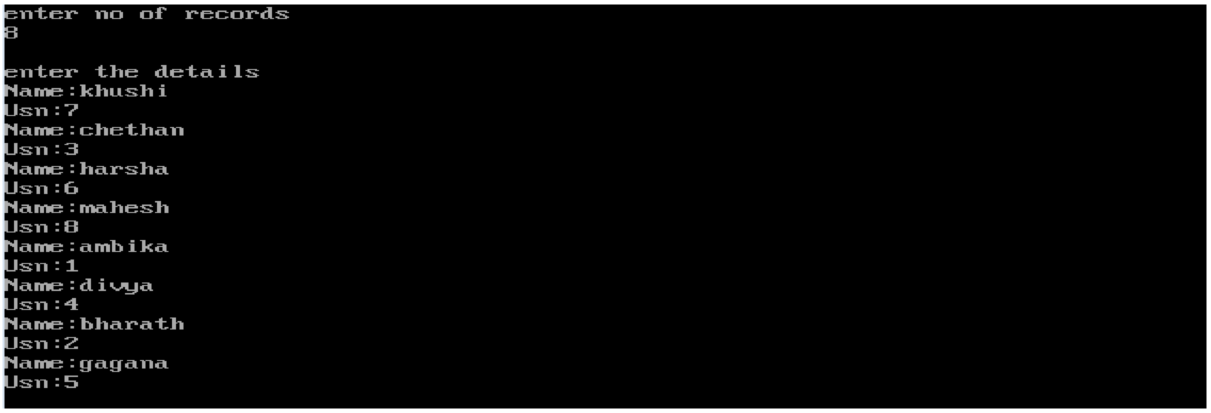
External sorting is a term for a class of sorting algorithms that can handle massive amounts of data. External sorting is required when the data being sorted do not fit into the main memory of a computing device (usually RAM) and instead they must reside in the slower external memory (usually a hard drive). External sorting typically uses a hybrid sort-merge strategy. In the sorting phase, chunks of data small enough to fit in main memory are read, sorted, and written out to a temporary file.k-way merge is the algorithm that takes as input k sorted arrays, each of size n. It outputs a single sorted array of all the elements.It does so by using the "merge" routine central to the merge sort algorithm to merge array 1 to array 2, and then array 3 to this merged array, and so on until all k arrays have merged

Algorithm:

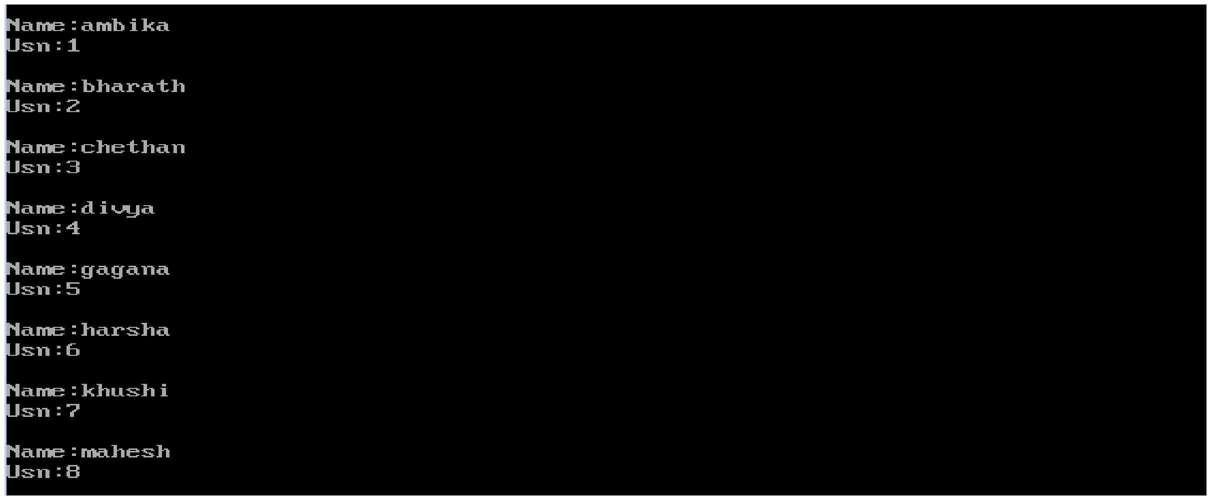
```
step1: Start
step 2: Enter the required number of records .
step 3: Create 8 file in write mode using
file[i].open(fname[i],ios::out);
step  4: Store one record in each files and close the file using
file[i].close();
step 5: call kwaymerge();
step 6: k=0
         char file[7][20]={“11.txt, “22.txt”,”33.txt”,”44.txt”,”111.txt”,”222.txt”,”1111.txt”}
step 7: for(i=0;i<8;i+=2)
        merge_file(fname[i],fname[i+1],filename[k++])
        goto step 10
step8 : k=4
        for(0;i<4;i+=2)
        merge_file(filename[i],filename[i+1],filename[k++])
        goto step 10
step 9: merge_file(filename[4],file[5],filename[6])
        goto step 10
step 10: merge_file(char*file1,char *file2,char *filename)
step 11: k=0
        fstream f1,f2;
step 12: open 2 files in read mode
        f1.open(file1,ios::in);
        f2.open(file2,ios::in);
step13: while(!f1.eof()) true goto 14 else goto 15
step 14: f1.getline(recd[k].nam,20,’|’);
        f1.getline(recd[k++].usn,20,’\n’);
        goto 13
step 15: while(!f2.eof()) true goto 16 else goto 17
step 16: f2.getline(recd[k].name,20,'|');
        f2.getline(recd[k++].usn,20,'\n');
        goto 15
step 17: for(t=0;<k-t-2;y++)
        for(y=0;y<k-t-2;y++)
        if(strcmp(recd[y].name,recd[y+1].name)>0)
        true  goto 18 else goto 19
step 18 :temp=recd[y];
        recd[y]=recd[y+1]
        recd[y+1]=temp
step 19: fstream temp1;
        temp1.open(filename,ios::out)
step 20: for(t=1;t<k-1;t++)
        temp1<<recd[t].name<<"|"<<recd[t].usn<<"\n";
step 21: f1.close();
        f2.close();
        temp 1.close();
        goto step 8 or 9 or step 22
step 22: fstream result;
`        result.open("1111.txt",ios::in);
Step 23: print the sorted records using
        for(i=0;i<no;i++)
        result.getline(name,20,'|')
        result.getline(usn,20,'\n')
step 24: stop.
```

Program:

```
#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
```



```
Name:ambika
Usn:1

Name:bharath
Usn:2

Name:chethan
Usn:3

Name:divya
Usn:4

Name:gagana
Usn:5

Name:harsha
Usn:6

Name:khushi
Usn:7

Name:mahesh
Usn:8
```