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.

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

int main()
{
   char name[10][20];
   char temp[20];
   char ifilename[15], ofilename[15];
   int num, i;
//   clrscr();
   cout << "Enter how many names you want to enter" << endl;
   cin >> num;
   cout << "Enter the names" << endl;
   for (i = 0; i < num; i++)
      cin >> name[i];
   cout << "Names in reverse order" << endl;
   for (i = num - 1; i >= 0; i--)
      cout << strrev(name[i]) << endl;
   cout << "Enter the input file name which contains a list of names" << endl;
   cin >> ifilename;
   cout << "Enter the output file name where the reversed list of names has to be written" <<
endl;
   cin >> ofilename;
   ifstream infile(ifilename);
   ofstream outfile(ofilename);
   if (!infile)
   {
      cout << "Could not open the specified input file" << endl;
      getch();
      exit(0);
   }
   while (infile.getline(temp, 20, '\n'))
   {
      outfile << strrev(temp) << endl;
   }
   outfile.close();
   infile.close();
   cout << "Output written to file " << ofilename << endl;
   getch();
   return 0;
}
```

OUTPUT:

```
Enter how many names you want to enter
3
Enter the names
keerthi
deepak
imran
Names in the reverse order
ihtreek
kapeed
narmi
Enter the input file name which contains list of names
input.txt
Enter the output file name where reversed list of names has to be written
output.txt
Output written to file output.txt_
```

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

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 << "Enter the USN: ";
    cin >> s.usn;
    cout << "Enter the age: ";
    cin >> s.age;
    cout << "Enter the sem: ";
    cin >> s.sem;
    cout << "Enter 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 the buffer
    file.close();
}

void search()
{
    char usn[10];
    char extra[45];
    file.open("student.txt", ios::in);
    if (!file)
    {
        cout << "Unable 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())
```

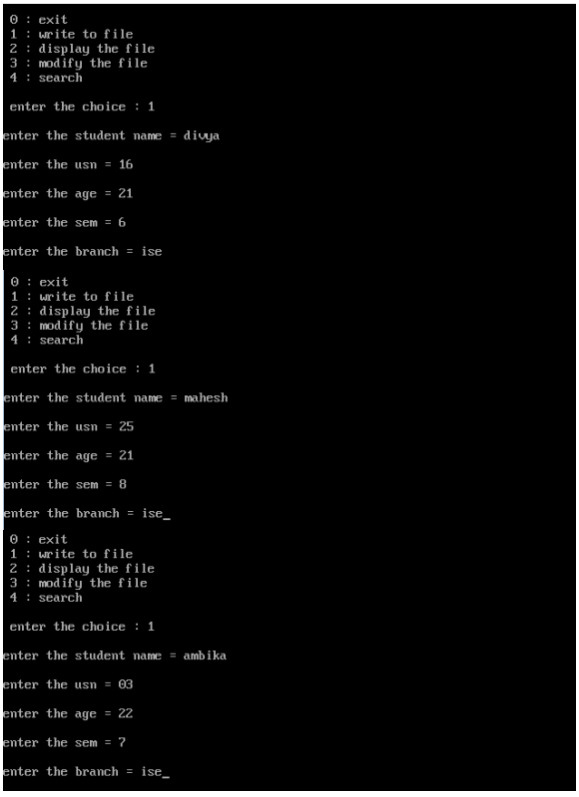
```
    {
        file.getline(s.name, 10, '|');
        file.getline(s.usn, 10, '|');
        file.getline(s.age, 5, '|');
        file.getline(s.sem, 5, '|');
        file.getline(s.branch, 5, '!');
        file.getline(extra, 45, '\n');

        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();
            return;
        }
    }

    cout << "\nRecord not found";
    file.close();
}

void displayFile()
{
    char extra[45];
    file.open("student.txt", ios::in);
    if (!file)
    {
        cout << "\nCannot open the file in read mode";
        exit(1);
    }

    cout << "\n\nNAME\t\tUSN\t\tAGE\t\tSEM\t\tBRANCH";
    cout << "\n----\t\t---\t\t---\t\t---\t\t------";

    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);
    }

    file.close();
}

void modify()
```

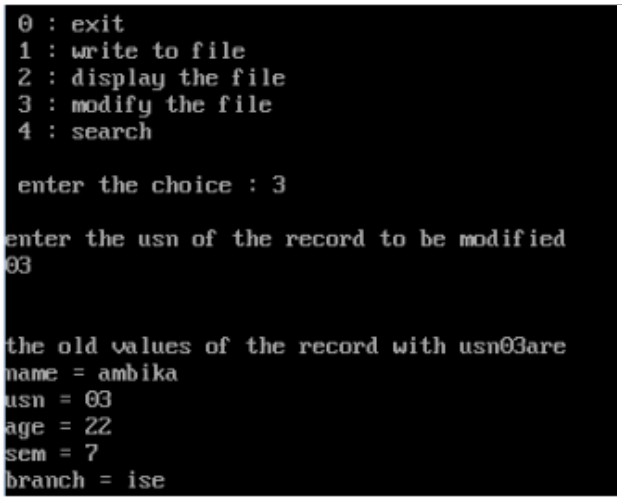
```
{
   char usn[10];
   char buffer[45];
   char extra[45];
   int i, j;
   Student s[20];

   file.open("student.txt", ios::in);
   if (!file)
   {
      cout << "\nUnable to open the file in input mode";
      exit(1);
   }

   cout << "\nEnter the USN of the record to be modified: ";
   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 << "The 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:";
         cout << "\nName: ";
         cin >> s[j].name;
         cout << "USN: ";
         cin >> s[j].usn;
         cout << "Age: ";
         cin >> s[j].age;
         cout << "Sem: ";
         cin >> s[j].sem;
```

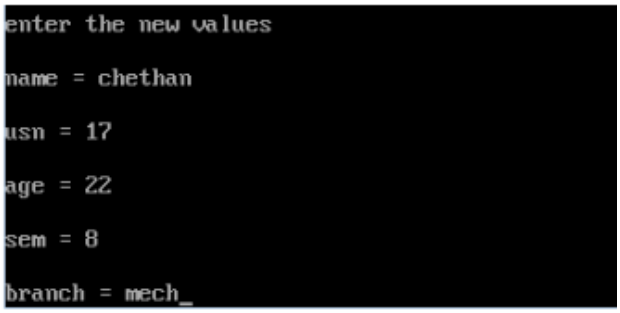
```
            cout << "Branch: ";
            cin >> s[j].branch;
            break;
        }
    }

    if (j == i)
    {
        cout << "\nThe record with USN " << usn << " is not present";
        return;
    }

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

int main()
{
    int choice;
    while (1)
    {
//      clrscr();
        cout << "\n0: Exit";
```

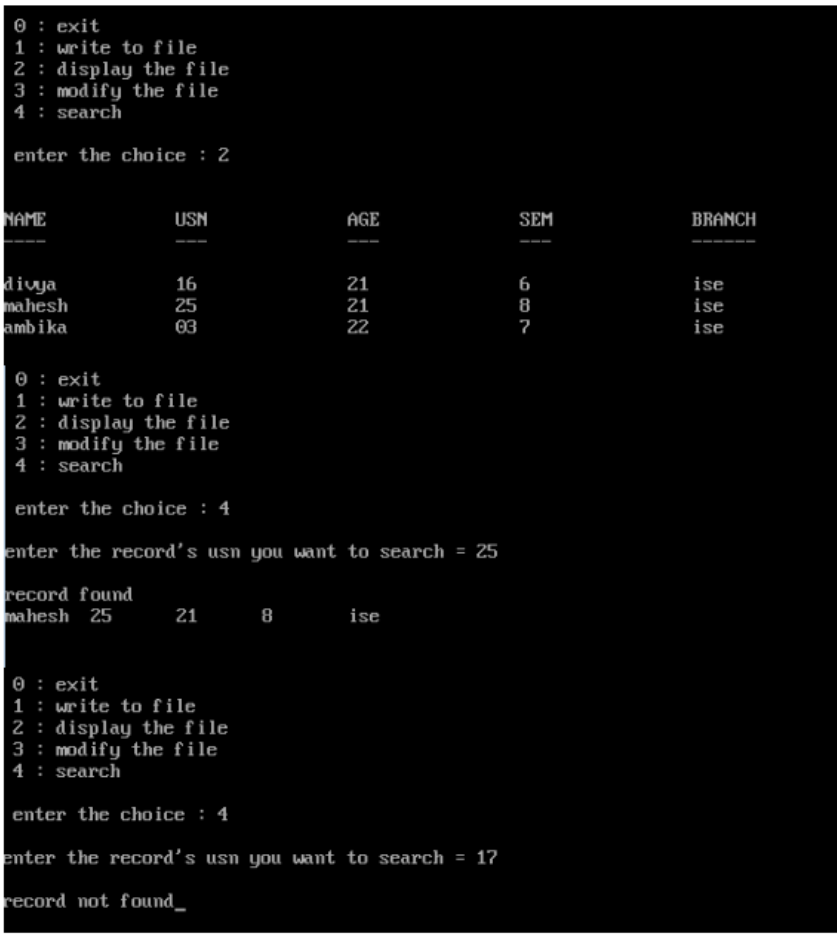
```
    cout << "\n1: Write to file";
    cout << "\n2: Display the file";
    cout << "\n3: Modify the file";
    cout << "\n4: Search";
    cout << "\n\nEnter the choice: ";
    cin >> choice;

    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;
    }
  }

  return 0;}
```

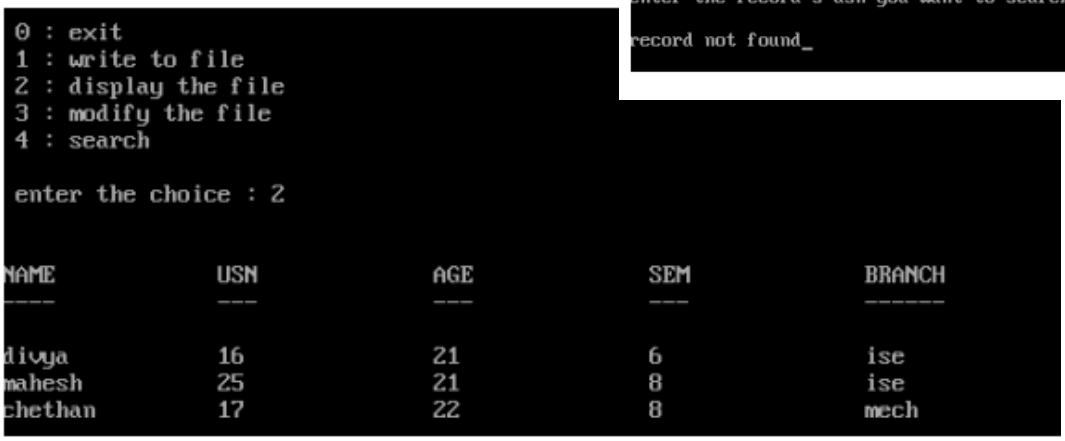
OUTPUT:











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 <iostream>
#include <fstream>
#include <cstring>

using namespace std;

class Student
{
public:
   string name;
   string usn;
   string age;
   string sem;
```

```
    string branch;
};

fstream file;
Student s;

void writeRecord()
{
    file.open("program_3.txt", ios::app);
    if (!file)
    {
        cout << "Cannot open the file in append mode";
        exit(1);
    }

    cout << "\nEnter the student name: ";
    cin >> s.name;
    cout << "Enter the USN: ";
    cin >> s.usn;
    cout << "Enter the age: ";
    cin >> s.age;
    cout << "Enter the sem: ";
    cin >> s.sem;
    cout << "Enter the branch: ";
    cin >> s.branch;

    file << s.name << "|" << s.usn << "|" << s.age << "|" << s.sem << "|" << s.branch <<
endl;
    file.close();
}

void search()
{
    string usn;
    file.open("program_3.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 (getline(file, s.name, '|'))
    {
        getline(file, s.usn, '|');
        getline(file, s.age, '|');
        getline(file, s.sem, '|');
        getline(file, s.branch);
```

```
        if (s.usn == usn)
        {
            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();
            return;
        }
    }

    cout << "\nRecord not found";
    file.close();
}

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

    cout << "\n\nNAME\t\tUSN\t\tAGE\t\tSEM\t\tBRANCH";
    cout << "\n---------------------------------------";

    while (getline(file, s.name, '|'))
    {
        getline(file, s.usn, '|');
        getline(file, s.age, '|');
        getline(file, s.sem, '|');
        getline(file, s.branch);

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

    file.close();
}

void modify()
{
    string usn;
    string newValues[4];
    file.open("program_3.txt", ios::in);
    if (!file)
    {
        cout << "\nUnable to open the file in input mode";
```

```
        exit(1);
    }

    cout << "\nEnter the USN: ";
    cin >> usn;

    ofstream tempFile("temp.txt");

    while (getline(file, s.name, '|'))
    {
        getline(file, s.usn, '|');
        getline(file, s.age, '|');
        getline(file, s.sem, '|');
        getline(file, s.branch);

        if (s.usn == usn)
        {
            cout << "\nThe old values of the record with USN " << usn << " are:";
            cout << "\nName: " << s.name;
            cout << "\nUSN: " << s.usn;
            cout << "\nAge: " << s.age;
            cout << "\nSem: " << s.sem;
            cout << "\nBranch: " << s.branch;
            cout << "\nEnter the new values:\n";
            cout << "Name: ";
            cin >> newValues[0];
            cout << "Age: ";
            cin >> newValues[1];
            cout << "Sem: ";
            cin >> newValues[2];
            cout << "Branch: ";
            cin >> newValues[3];
            tempFile << newValues[0] << "|" << s.usn << "|" << newValues[1] << "|" <<
newValues[2] << "|" << newValues[3] << endl;
        }
        else
        {
            tempFile << s.name << "|" << s.usn << "|" << s.age << "|" << s.sem << "|" <<
s.branch << endl;
        }
    }

    tempFile.close();
    file.close();

    remove("program_3.txt");
    rename("temp.txt", "program_3.txt");
}

int main()
```

```
{
    int choice;
    while (true)
    {
        cout << "\n0: Exit";
        cout << "\n1: Write to file";
        cout << "\n2: Display the file";
        cout << "\n3: Modify the file";
        cout << "\n4: Search";
        cout << "\n\nEnter 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;
        }
    }

    return 0;
}
```

```
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_
```

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 <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <sstream>

using namespace std;

class Student
{
public:
    string name;
    string usn;
```

```
    string age;
    string sem;
    string branch;
    string buffer;
};

void writeRecord()
{
    ofstream file;
    Student s;
    int k, rrn = 0, n;
    file.open("program_4.txt", ios::app);
    if (!file)
    {
        cout << "\nCannot open the file in append mode\n";
        exit(0);
    }
    cout << "How many records: ";
    cin >> n;
    for (k = 0; k < n; k++)
    {
        cout << "\nEnter the student name: ";
        cin >> s.name;
        cout << "Enter the usn: ";
        cin >> s.usn;
        cout << "Enter the age: ";
        cin >> s.age;
        cout << "Enter the sem: ";
        cin >> s.sem;
        cout << "Enter the branch: ";
        cin >> s.branch;
        file << rrn << "|" << s.name << "|" << s.usn << "|"
             << s.age << "|" << s.sem << "|" << s.branch << "\n";
        rrn++;
    }
    file.close();
}

void displayFile()
{
    ifstream file;
    Student s;
    string rrn;
    int rrnValue = 0;
    file.open("program_4.txt");
    if (!file)
    {
        cout << "\nCannot open the file in input mode\n";
        exit(1);
    }
```

```
    cout << "\nRRN\tName\t\tUSN\t\tAge\t\tSem\t\tBranch\n";
    while (getline(file, s.buffer))
    {
        stringstream ss(s.buffer);
        getline(ss, rrn, '|');
        getline(ss, s.name, '|');
        getline(ss, s.usn, '|');
        getline(ss, s.age, '|');
        getline(ss, s.sem, '|');
        getline(ss, s.branch);
        cout << rrnValue << "\t" << s.name << "\t\t" << s.usn << "\t\t" << s.age << "\t\t" <<
s.sem << "\t\t" << s.branch << endl;
        rrnValue++;
    }
    file.close();
}


void search()
{
    ifstream file;
    Student s;
    string rrn;
    int rrnValue = 0;
    int searchRRN;
    file.open("program_4.txt");
    if (!file)
    {
        cout << "\nCannot open the file in input mode";
        exit(0);
    }
    cout << "\nEnter the RRN to be searched: ";
    cin >> searchRRN;
    cout << "\nRRN\tName\t\tUSN\t\tAge\t\tSem\t\tBranch\n";
    while (getline(file, s.buffer, '|'))
    {
        rrn = s.buffer;
        getline(file, s.name, '|');
        getline(file, s.usn, '|');
        getline(file, s.age, '|');
        getline(file, s.sem, '|');
        getline(file, s.branch, '\n');
        if (rrnValue == searchRRN)
        {
            cout << rrnValue << "\t" << s.name << "\t\t" << s.usn << "\t\t" << s.age << "\t\t" <<
s.sem << "\t\t" << s.branch << endl;
            cout << "\nRecord found\n";
            file.close();
            return;
        }
```

```
        rrnValue++;
    }
    cout << "\nRecord not found\n";
    file.close();
}

int main()
{
    int choice;
    while (1)
    {
        cout << "\n0: Exit";
        cout << "\n1: Insert";
        cout << "\n2: Search";
        cout << "\n3: Display";
        cout << "\nEnter the choice: ";
        cin >> choice;
        switch (choice)
        {
        case 1:
            writeRecord();
            break;
        case 2:
            search();
            break;
        case 3:
            displayFile();
            break;
        case 0:
            exit(0);
        default:
            cout << "\nInvalid option";
            break;
        }
    }
    return 0;}
```

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_
```

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<iostream>
#include<fstream>
#include<string>
using namespace std;
int n;
string usn_list[100];
int addr_list[100];
int cnt;
class student
{
        public:
                string usn,name,sem;
                void add_rec(fstream &);
                void get_data();
```

```
};
void student::get_data()
{
	cout<<"\nUSN : ";
	cin>>usn;
	cout<<"\nName : ";
	cin>>name;
	cout<<"\nSem : ";
	cin>>sem;
}
void create_index()
{
	void sort_index();
	int pos;
	string buf,urn;
	fstream fp("inp.txt",ios::in);
	cnt=-1;
	while(fp)
	{
		pos=fp.tellg();
		buf.erase();
		getline(fp,buf);
		int i=0;
		if(buf[i]=='*')
			continue;
		urn.erase();
		while(buf[i]!='|')
			urn+=buf[i++];
		usn_list[++cnt]=urn;
		addr_list[cnt]=pos;
	}
	fp.close();
	sort_index();
	for(int i=0;i<cnt;i++)
		cout<<usn_list[i]<<'|'<<addr_list[i]<<'\n';
}
void sort_index()
{
	int t_addr;
	string t_usn;
	cout<<cnt<<'\n';
	for(int i=0;i<cnt-1;i++)
	{
		for(int j=0;j<cnt-1-i;j++)
		{
			if(usn_list[j]>usn_list[j+1])
			{
				t_usn=usn_list[j];
				usn_list[j]=usn_list[j+1];
				usn_list[j+1]=t_usn;
```

```
                        t_addr=addr_list[j];
                        addr_list[j]=addr_list[j+1];
                        addr_list[j+1]=t_addr;
                    }
                }
        }
}

void student::add_rec(fstream &fp)
{
        fp.seekp(0,ios::end);
        fp<<usn<<'|'<<name<<'|'<<sem<<"\n";
}

int search( string key)
{
  int pos=0,adr,l=0,h=cnt,mid,flag=0;
  string buffer;
  fstream fp("inp.txt",ios::in);
  while(l<=h)
  {
        mid=(l+h)/2;
        if(usn_list[mid]==key)
        {
                flag=1;
                break;
        }
        if(usn_list[mid]>key)
                h=mid-1;
        if(usn_list[mid]<key)
                l=mid+1;
  }
        if(flag)
        {
                adr=addr_list[mid];
                fp.seekp(adr,ios::beg);
                getline(fp,buffer);
                cout<<"\nFond the record "<<buffer;
                cout<<' ' <<mid<<"mid\n";
                return mid;
        }
        else
        {
                cout<<"\nNot found";
                return -1;
        }
}

void del_rec(string key)
{
```

```
        int pos,adr;
        fstream fp;
        pos=search(key);
        adr=addr_list[pos];
        if(pos !=-1)
        {
                fp.open("inp.txt",ios::out | ios::in);
                fp.seekp(adr,ios::beg);
                fp.put('*');
                cout<<"\nRecord added!";
                fp.close();
                for(int i=pos;i<cnt;i++)
                {
                        usn_list[i]=usn_list[i+1];
                        addr_list[i]=addr_list[i+1];
                }
                cnt--;

        }
        else
                cout<<"\n Record not found!";
}
int main()
{
        student s[100];
        string key;
        fstream fp;
        for(;;)
        {
        int ch;
                cout<<"\nenter ur choice \n1.add rec\n2. show index\n3.search\n4. delete\n5.
exit";
                cin>>ch;
                switch(ch)
                {
                        case 1:
                                fp.open("inp.txt", ios::out);
                                cout<<"enter how many records\n";
                                cin>>n;
                                for(int i=0; i<n; i++)
                                {
                                        s[i].get_data();
                                        s[i].add_rec(fp);
                                }
                                fp.close();
                                break;

                        case 2: create_index();
                                break;
```

```
                case 3: cout<<"enter key of record to searched\n";
                        cin>>key;
                        search(key);
                        break;

                case 4: cout<<"enter key of record to deleted\n";
                        cin>>key;
                        del_rec(key);
                        break;

                case 5: exit(0);
            }
        }
        return 0 ;
}
```

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_
```

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<string>
#include<cstring>
#include<fstream>
#include<iomanip>
#include<iostream>

using namespace std;

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

char st_no[5] , rt_name[20];
int no;
void sort()
{
   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 create_index_file()
{
   ofstream index , index1;
   int i;
   index.open("secindex.txt" , ios::out);
   index1.open("record.txt" , ios::out);
   for( i = 0; i < no; i++)
   {
      if(i == no-1)
      {
         index <<rec[i].name<<"|"<<rec[i].usn<<"|"<<i+1;
         index1 <<i+1<<"|"<<rec[i].usn<<"|"<<rec[i].name<<"|"<<rec[i].sem;
```

```
        }

        else
        {
            index <<rec[i].name<<"|"<<rec[i].usn<<"|"<<i+1<<endl;
            index1 <<i+1<<"|"<<rec[i].usn<<"|"<<rec[i].name<<"|"<<rec[i].sem<<endl;
        }

    }
    index.close();
    index1.close();
}

void retrieve_record(char *index)
{
    fstream f1;
    int i;
    char buff[80],*p;
    f1.open("record.txt",ios::in);
    while(!f1.eof())
    {
        f1.getline(buff,80,'\n');
        p=strtok(buff,"|");
        if(strcmp(index, p)==0)
        {
            cout<<"\n\nStudent Details\n";
            cout<<"\nUSN\t\tName\tSemester\n";
            while(p!=NULL)
            {
                p=strtok(NULL,"|");
                if(p!=NULL)
                cout<<p<<"\t";
            }
        }

    }
    f1.close();

}
void delete_record(char *idx)
{
    fstream f1;
    int i;
    char buff[80],*p,index[20][20];
    f1.open("record.txt",ios::in);
    i=0;
    while(!f1.eof())
    {
        f1.getline(buff,80,'\n');
        p=strtok(buff,"|");
```

```
        strcpy(index[i],p);
        p=strtok(NULL,"|");
        strcpy(rec[i].usn,p);
        p=strtok(NULL,"|");
        strcpy(rec[i].name,p);
        p=strtok(NULL,"|");
        strcpy(rec[i].sem,p);
        i++;

    }
    no=i;
    f1.close();
    int k=-1;
    for(i=0;i<no;i++)
    {
        if(strcmp(index[i],idx)==0)
        {
            k=i;
            break;
        }
    }
    if(k>-1)
    {
        for(i=k;i<no-1;i++)
        {
            rec[i]=rec[i+1];
        }
        no--;
        sort();
        create_index_file();
        cout<<"\nData Successfully Deleted\n";

    }
    else
    {
        cout<<"\nInvalid Name\n";
    }

}

void display_record()
{
    char buff[80] , *p;
    int flag=1;
    ifstream f1;
    f1.open("record.txt" , ios::in);
    cout<<"\n\nStudent Details\n";
    cout<<"USN\t\tName\tSemester\n";
    while(! f1.eof())
    {
```

```
        f1.getline(buff , 80 , '\n');
        p= strtok(buff, "|");
        while(p!= NULL)
        {
            flag =0;
            p= strtok(NULL , "|");
            if(p != NULL)
                cout<<p<<setw(15);
        }
        cout<<endl<<setw(0);
    }

    if(flag == 1)
        cout<<"\nNo record found";
    f1.close();
}

void retrieve_details(int ch)
{
    int k=0, i;
    char buff[80] , *p;
    ifstream f1;
    char chusn[20] , index[20][80];
    f1.open("secindex.txt" , ios::in);
    while(!f1.eof())
    {
        f1.getline(buff , 80 , '\n');
        p = strtok(buff , "|");
        if(strcmp(rt_name , p) == 0)
        {
            strcpy(found[k].name , p);
            p = strtok(NULL , "|");
            strcpy(found[k].usn , p);
            p = strtok(NULL , "|");
            strcpy(index[k] , p);
            k++;
        }
    }

    if(k == 1)
    {
        if(ch == 2)
            retrieve_record(index[0]);
        else
            delete_record(index[0]);
    }
    else if(k > 1)
    {
        cout<<"Please choose the candidate USN\n";
        for( i = 0; i < k; i++)
```

```
        {
            cout<<"Name = "<<found[i].name <<"USN = "<<found[i].usn<<endl;
        }
        cin>>chusn;
        for(i=0; i<k ; i++)
        {
            if(strcmp(chusn , found[i].usn) == 0)
            {
                if(ch == 2)
                    retrieve_record(index[i]);
                else
                    delete_record(index[i]);
            }
        }

    }
    else
        cout<<"Invalid Name\n";

}

int main()
{
    int ch, flag=1;
    while(flag)
    {
        cout<<"\n1. Add New records\n2.Retrieve Record\n3.Delete a
Record\n4.Display\n5.Exit\n";
        cout<<"Enter the choice\n";
        cin>>ch;
        switch (ch)
        {
            case 1: cout<<"Enter the Number of record\t";
                    cin>>no;
                    for(int i = 0; i < no; i++)
                    {
                        cout<<"Enter the details of "<<i+1<<"th student";
                        cout<<"\nUSN\t";
                        cin>>rec[i].usn;
                        cout<<"\nName\t";
                        cin>>rec[i].name;
                        cout<<"\nSem\t";
                        cin>>rec[i].sem;
                    }
                    sort();
                    create_index_file();
                    break;
            case 2:
            case 3: if(ch ==2)
                        cout<<"Enter the name to search\t";
```

```
            else
               cout<<"Enter the student name to delete\t";
            cin>>rt_name;
            retrieve_details(ch);
            break;
        case 4: display_record();
            break;

        default:
            flag =0;
            break;
      }
   }
   return 0;
}
```

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
```

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 <iostream>
#include <fstream>
#include <iomanip>
#include <cstring>

using namespace std;

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

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

   if (!out1 || !out2)
   {
      cout << "Unable to open one of the list files\n";
      return;
   }

   cout << "Enter the number of names you want to enter in file1: ";
   cin >> m;

   cout << "\nEnter the names in ascending order:\n";
   for (int i = 0; i < m; i++)
   {
      cin >> name;
      out1 << name << '\n';
   }

   cout << "\nEnter the number of names you want to enter in file2: ";
   cin >> n;

   cout << "\nEnter the names in ascending order:\n";
   for (int i = 0; i < n; i++)
   {
      cin >> name;
      out2 << name << '\n';
   }

   out1.close();
   out2.close();
}

int main()
```

```
{
   char list1[100][20];
   char list2[100][20];
   int m, n;

   writeLists();

   ifstream in1("file1.txt");
   ifstream in2("file2.txt");
   ofstream out3("file3.txt");

   if (!in1 || !in2 || !out3)
   {
      cout << "Unable to open one of the files\n";
      return 0;
   }

   cout << "\nLIST-1 CONTENTS\n";
   m = 0;
   while (in1.getline(list1[m], 20, '\n'))
   {
      cout << list1[m] << '\n';
      m++;
   }

   cout << "\nLIST-2 CONTENTS\n";
   n = 0;
   while (in2.getline(list2[n], 20, '\n'))
   {
      cout << list2[n] << '\n';
      n++;
   }

   m--;
   n--;

   cout << "\nElements common to both files are:\n";
   for (int i = 0; i < m; i++)
   {
      for (int j = 0; j < n; j++)
      {
         if (strcmp(list1[i], list2[j]) == 0)
         {
            out3 << list1[i] << '\n';
            cout << list1[i] << '\n';
         }
      }
   }

   in1.close();
```

```
    in2.close();
    out3.close();

    return 0;
}
```

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 <fstream>
#include <iostream>
#include <string>
#include <cstring>

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

std::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;
   std::fstream f1, f2;
   f1.open(file1, std::ios::in);
   f2.open(file2, std::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 (std::strcmp(recd[y].name, recd[y + 1].name) > 0) {
            temp = recd[y];
            recd[y] = recd[y + 1];
            recd[y + 1] = temp;
         }
   std::fstream temp1;
   temp1.open(filename, std::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;
    std::cout << "Enter the number of records: ";
    std::cin >> no;
    std::cout << "\nEnter the details:\n";
    for (i = 0; i < 8; i++)
        file[i].open(fname[i], std::ios::out);
    record rec;
    for (i = 0; i < no; i++) {
        std::cout << "Name: ";
        std::cin >> rec.name;
        std::cout << "USN: ";
        std::cin >> rec.usn;
        file[i % 8] << rec.name << "|" << rec.usn << "\n";
    }
    for (i = 0; i < 8; i++)
        file[i].close();
    kwaymerge();
    std::fstream result;
    result.open("1111.txt", std::ios::in);
    std::cout << "\nSorted records:\n";
    char name[20], usn[20];
    for (i = 0; i < no; i++) {
        result.getline(name, 20, '|');
        result.getline(usn, 20, '\n');
        std::cout << "\nName: " << name << "\nUSN: " << usn << "\n";
    }
    result.close();

    std::cin.ignore();
    std::cin.get();
    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
```