ACHARYA INSTITUTE OF TECHNOLOGY

ACHARYA DR. SARVEPALLI RADHAKRISHNAN ROAD,SOLADEVANAHALLI,

BANGALURU –560107

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

FILE STRUCTURE LABORATORY WITH MINI PROJECT

COMPILED BY

PROF. DHANANJAYA M K

ASSISTANT PROFESSOR

SHWETHA G N

MAMATHA G K

LAB INSTRUCTOR.

NAME: _____

USN: _____

SECTION: _____

FILE STRUCTURES LABORATORY

Subject Code: 18ISL67 **I.A. Marks : 40**
Hours/Week : 03 **Exam Hours: 03**
Total Hours : 42 **Exam Marks: 100**

Design, develop, and implement the following programs

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.

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

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.

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.

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.

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.

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

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

****

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

I/O redirection

Operating systems provide shortcuts for switching between standard I/O(stdin and stdout) and regular file I/O. I/O redirection is used to change a program so it writes its output to a regular file rather than to stdout.

- In both DOS and UNIX, the standard output of a program can be redirected to a file with the **>** symbol.
- In both DOS and UNIX, the standard input of a program can be redirected to a file with the **<** symbol.
- The notations for input and output redirection on the command line in Unix are

```
< file                (redirect stdin to "file")
> file                (redirect stdout to "file")
```

- Example:

```
list.exe > myfile
```

The output of the executable file is redirected to a file called “myfile”

pipe

- Piping: using the output of one program as input to another program. A connection betweenstandard output of one process and standard input of a second process.
- In both DOS and UNIX, the standard output of one program can be piped (connected) to thestandard input of another program with the **|** symbol.
- Example:

```
program1 | program2
```

- Output of program1 is used as input for program2

File I/O: perform output and input of characters to or from files

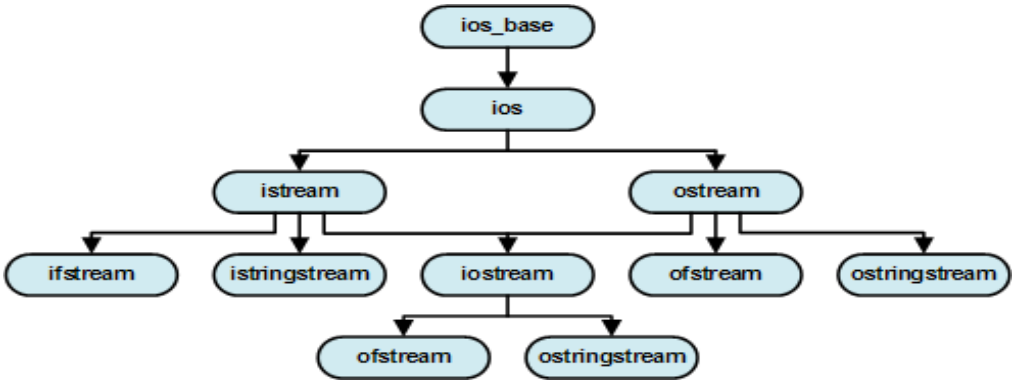
Standerd I/O:

- standard streams are preconnected input and output channels between a computer program and itsenvironment (typically a text terminal) when it begins execution. The three I/O connections are called standard input (stdin), standard output (stdout) and standard error (stderr).

fstream

- fstream provides an interface to read and write data from files as input/output streams.The file tobe associated with the stream can be specified either as a parameter in the constructor or by callingmember open.

- After all necessary operations on a file have been performed, it can be closed (or disassociated) by calling member close. Once closed, the same file stream object may be used to open another file.



Function to open a file:

The first operation generally performed on an object of one of these classes is to associate it to a real file. This procedure is known as to *open a file*. An open file is represented within a program by a stream object (an instantiation of one of these classes, in the previous example this was myfile) and any input or output operation performed on this stream object will be applied to the physical file associated to it.

In order to open a file with a stream object we use its member function open():

```
open (filename, mode);
```

Where filenameis a null-terminated character sequence of type const char *(the same type that string

literals have) representing the name of the file to be opened, and mode is an optional parameter with a combination of the following flags:

ios::in	Open for input operations.
ios::out	Open for output operations.
ios::binary	Open in binary mode.
ios::ate	Set the initial position at the end of the file. If this flag is not set to any value, the initial position is the beginning of the file.
ios::app	All output operations are performed at the end of the file, appending the content to the current content of the file. This flag can only be used in streams open for output-only operations.
ios::trunc	If the file opened for output operations already existed before, its previous content is deleted and replaced by the new one.

Function to Closing the Files

To disassociate a logical program file from a physical system file.

- **Prototypes:**

  int close (int Handle);

- **Example:**

  close (Input);

Getline Function:

Extracts characters from specified location and stores them into *str* until the delimitation character *delim* is found or length equal to size.

- **prototype**

  fstream str;
  Str.getline (istream& is, int size, char delim);

PROGRAM 1:

```
#include<fstream.h>
#include<iostream.h>
#include<string.h>
#include<conio.h>
int main()
{
```

```
        char ifilename[15], ofilename[15], temp[20], name[10][20];
        int num,i=0;
        fstream ifile,ofile;
        cout<<"Enter how many person details you want to enter: ";
        cin>>num;
        cout<<"Enter the file name where it has to be written: ";
        cin>>ofilename;
        ifile.open(ofilename,ios::out);
        cout<<"Enter person details: ";
        while(i<num)
        {
                cin>>name[i];
                ifile<<name[i]<<endl;
                i++;
        }
        ifile.close();
        cout<<"Enter the file name where output has to be written: \n ";
        cin>>ifilename;
        ifile.open(ofilename,ios::in);
        ofile.open(ifilename,ios::out);
        cout<<"The details of the person in reverse that are entered is:\n";
        while( ifile >> temp != NULL)
        {
                cout<<strrev(temp)<<endl;
                ofile<<temp<<endl;
        }
        getch();
        return 0;
}
```

OUTPUT:-

```
1: file i/o
2: standard i/o
0 : any other to
exit1
enter the number of names to
read2
Enter the
names
keerthana
madhu
The reversed names
areanahtreek
uhdam
```

1: file i/o
2: standard i/o
0: any other to
exit2
Enter the filename which contain list of
namesstudent.txt
Enter the filename to store names in reverse
orderstudentr.txt

1:file i/o
2:standard i/o

any other to exit
0
c:\tc>type student.txt
ragu
navya

c:\tc>type studentr.dat
ugar
ayvan

****

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

Fixed length record

A record which is predetermined to be the same length as the other records in the file.

Record 1	Record 2	Record 3	Record 4	Record 5

- The file is divided into records of equal size.
- All records within a file have the same size.
- Different files can have different length records.
- Programs which access the file must know the record length.
- Offset, or position, of the nth record of a file can be calculated.
- There is no external overhead for record separation.
- There may be internal fragmentation (unused space within records.)
- There will be no external fragmentation (unused space outside of records) except for deletedrecords.
- Individual records can always be updated in place

Delimited Variable Length Fields

Record 1	|	Record 2	|	Record 3	|	Record 4	|	Record 5

- The fields within a record are followed by a delimiting byte or series of bytes.
- Fields within a record can have different sizes.
- Different records can have different length fields.
- Programs which access the record must know the delimiter.
- The delimiter cannot occur within the data.
- If used with delimited records, the field delimiter must be different from the record delimiter.
- There is external overhead for field separation equal to the size of the delimiter per field.
- There should be no internal fragmentation (unused space within fields.)

Pack():

This method is used to group all the related field values of particular record taken by the application in buffer.

Unpack():

This method is used to ungroup all the related field values of percular record taken from the file in buffer.

PROGRAM 2:

```
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
#include<string.h>
#include<stdio.h>
#include<conio.h>
#define SIZE 50
fstream file;

class fixedf
{
        struct student
    {
        char usn[11], name[15], sem[6], dept[6];

    };
    public: void pack();
        void unpack();
        void search();
        void modify();
};

void fixedf::pack()
{
    char b[SIZE+1];
    student s;
    cout<<"\nEnter usn, name, sem, dept: ";
    cin>>s.usn>>s.name>>s.sem>>s.dept;
    file.open("student.txt",ios::app);
    sprintf(b,"%s|%s|%s|%s|",s.usn,s.name,&s.sem,s.dept);
    int  len=strlen(b);
    while(len<(SIZE))
    {
        strcat(b,"-");
        len++;
    }
    strcat(b,"\n");
    file<<b;
    file.close();
}


void fixedf::search()
{
```

```
        char b[SIZE+1], usn[11];
        student s;
        file.open("student.txt",ios::in);
        cout<<"\nEnter usn to be searched: ";
        cin>>usn;
        while(!file.eof())
        {
                file.getline(b,100,'\n');
                sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]",s.usn,s.name,s.sem,s.dept);
                if(strcmp(s.usn,usn)==0)
                {
                        cout<<"\nRecord found\n";
                        cout<<s.usn<<" "<<s.name<<" "<<s.sem<<" "<<s.dept<<endl;
                        return;
                }
        }
        cout<<"\n Record not found";
        return;
}

void fixedf::unpack()
{
        char b[SIZE+1];
        student s;
        file.open("student.txt",ios::in);
        while(!file.eof())
        {
                file.getline(b,52,'\n');
                if(file.eof())
                return;
                sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]",s.usn,s.name,s.sem,s.dept);
                cout<<s.usn<<" "<<s.name<<" "<<s.sem<<" "<<s.dept<<endl;
        }
}

void fixedf::modify()
{       char b[SIZE+1],usn[11];
        student s;
        int n=0;
                file.open("student.txt",ios::in|ios::out);
        cout<<"\nEnter usn to be modified: ";
        cin>>usn;
        while(!file.eof())
        {
                file.getline(b,100,'\n');
                sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]",s.usn,s.name,s.sem,s.dept);
                if(strcmp(s.usn,usn)==0)
```

```
                {
                        cout<<"\nKey found\n: ";
                        cout<<"\nENter USN: "; cin>>s.usn;
                        cout<<"\nENter name: "; cin>>s.name;
                        cout<<"\nENter sem: "; cin>>s.sem;
                        cout<<"\nEnter dept: "; cin>>s.dept;

                        sprintf(b,"%s|%s|%s|%s|",s.usn,s.name,s.sem,s.dept);
                        int  len=strlen(b);
                        while(len<(SIZE))
                        {
                                strcat(b,"-");
                                len++;
                        }
                        strcat(b,"\n");
                        file.seekp((n*(SIZE+2)),ios::beg);
                        file<<b;
                        return;
                }
                n++;
        }
        cout<<"\n Record not found";
        return;
}
void main()
{
        fixedf f;
        int ch;
        clrscr();
        for(;;)
        {
                cout<<"\n1: pack 2: unpack 3: search 4: Modify 5:Exit\nEnter your choice: ";
                cin>>ch;
                switch(ch)
                {
                        case 1: f.pack();
                                break;
                        case 2: f.unpack();
                                file.close();
                                break;
                        case 3: f.search();
                                file.close();
                                break;
                        case 4: f.modify();
                                file.close();
                                break;
                        default: exit(0);
```

```
            }
        getch();
        }}
```

OUTPUT:

Output :

1:write to file 2:display the file 3:modify the file 4:search 5.exit

```
Enter the choice:1
Enter the number of students:2

Enter the student name = ajay
Enter the sem = 6
Enter the branch = ise

Enter the student name = rahul
Enter the sem = 6
Enter the branch = cse
```

1:write to file 2:display the file 3:modify the file 4:search 5.exit

```
Enter the choice:2
Name              Sem         Branch
ajay              6           ise||||||||||
rahul             6           cse||||||||||
```

1:write to file 2:display the file 3:modify the file 4:search 5.exit

```
Enter the choice:4
Enter the record name you want to search = rahul
Record found
rahul       6           cse||||||||||
```

1:write to file 2:display the file 3:modify the file 4:search 5.exit

```
Enter the choice:3
Enter the record name you want to modify:rahul
record found and details are:
rahul       6           cse||||||||||
enter modification details
Enter the student name =navya
Enter the sem = 6
Enter the branch = ise
```

1:write to file 2:display the file 3:modify the file 4:search 5.exit

```
Enter the choice:2
Name              Sem         Branch
ajay              6           ise||||||||||
Navya             6           ise||||||||||
```

1:write to file 2:display the file 3:modify the file 4:search 5.exit

```
Enter the choice:4
Enter the record name you want to search:keerthi
```

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

Variable length record

A record which can differ in length from the other records of the file.

- **delimited record**

A variable length record which is terminated by a special character or sequence of characters.

- **delimiter**

  A special character or group of characters stored after a field or record, which indicates the end of the preceding unit.

  - The records within a file are followed by a delimiting byte or series of bytes.
  - The delimiter cannot occur within the records.
  - Records within a file can have different sizes.
  - Different files can have different length records.
  - Programs which access the file must know the delimiter.
  - Offset, or position, of the nth record of a file cannot be calculated.
  - There is external overhead for record separation equal to the size of the delimiter per record.
  - There should be no internal fragmentation (unused space within records.)
  - There may be no external fragmentation (unused space outside of records) after file updating.
  - Individual records cannot always be updated in place.

PROGRAM 3:

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

fstream file,file1;

class variable
{
        struct student
        {
                char usn[20],name[20],sem[20],dept[20];
```

```
        };
        public: void pack();
                void unpack();
                void search();
                void modify();
};

void variable::pack()
{
        char b[100];
        student s;
        cout<<"\nEnter usn, name, sem, dept: ";
        cin>>s.usn>>s.name>>s.sem>>s.dept;
        file.open("student1.txt",ios::app);
        sprintf(b,"%s|%s|%s|%s$",s.usn,s.name,s.sem,s.dept);
        file<<b;
        file.close();
}


void variable::unpack()
{
        char b[100];
        student s;
        file.open("student1.txt",ios::in);
        cout<<"\nDatabase: \n";
        while(!file.eof())
        {
                file.getline(b,100,'$');
                if(file.eof())
                break;
                sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]$",s.usn,s.name,s.sem,s.dept);
                cout<<s.usn<<" "<<s.name<<" "<<s.sem<<" "<<s.dept<<endl;
        }
        file.close();
        getch();
}

void variable::search()
{
        char b[100],usn[11];
        int flag=0;
        student s;
        file.open("student1.txt",ios::in);
        cout<<"\nEnter usn to be searched: ";
        cin>>usn;
        while(1)
```

```
	{
		file.getline(b,100,'$');
		if(file.eof())
			break;
		sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]$",s.usn,s.name,s.sem,s.dept);
		if(strcmp(s.usn,usn)==0)
		{
			flag=1;
			cout<<s.usn<<" "<<s.name<<" "<<s.sem<<" "<<s.dept<<endl;
		}
	}
	if(!flag0)
		cout<<"\nKey not found";
	file.close();
}

void variable::modify()
{
	char b[100],usn[11];
	student s;
	int flag=0;
	file1.open("temp.txt",ios::app);
	file.open("student1.txt",ios::in);
	cout<<"\nEnter usn to be modified: ";
	cin>>usn;
	while(1)
	{
		file.getline(b,100,'$');
		if(file.eof())
		break;
		sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]$",s.usn,s.name,s.sem,s.dept);
		if(strcmp(s.usn,usn)==0)
		{
			cout<<"\nKey found\n: ";
			cout<<"\nEnter USN:"; cin>>s.usn;
			cout<<"\nENter name: "; cin>>s.name;
			cout<<"\nENter sem: "; cin>>s.sem;
			cout<<"\nEnter dept: "; cin>>s.dept;
			sprintf(b,"%s|%s|%s|%s$",s.usn,s.name,s.sem,s.dept);
			file1<<b;
			flag=1;
		}
		else
		{
			strcat(b,"$");
			file1<<b;
		}
```

```
        }
        file1.close();
        file.close();
        remove("student1.txt");
        rename("temp.txt","student1.txt");
        if(!flag)
                printf("record not found");
}

void main()
{
        variable f;
        int ch;
        for(;;)
        {
                cout<<"\n1: pack 2: unpack 3: search 4: Modify 5:Exit\nEnter your choice: ";
                cin>>ch;
                switch(ch)
                {
                        case 1:f.pack();
                             break;
                        case 2:f.unpack();
                             break;
                        case 3:f.search();
                             break;
                        case 4:f.modify();
                             break;
                        default: exit(0);
                }
        }
}
```

OUTPUT:

```
 1:write to file 2:display the file 3:modify the file 4:search 5.exit
 Enter the choice:1
 Enter the number of students:2
 Enter the student name = ajay
 Enter the sem = 6
 Enter the branch = ise

 Enter the student name = rahul
 Enter the sem = 6
 Enter the branch = cse

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

Enter the choice:2
 Name              Sem         Branch
```

```
ajay              6           ise
rahul             6           cse

1:write to file 2:display the file 3:modify the file 4:search 5.exit
Enter the choice:4
Enter the record name you want to search = rahul
Record found
rahul       6           cse

1:write to file 2:display the file 3:modify the file 4:search 5.exit
Enter the choice:3
Enter the record name you want to modify:rahul
record found and details are:
rahul       6           cse

enter modification details
Enter the student name =navya
Enter the sem = 6
Enter the branch = ise
1:write to file 2:display the file 3:modify the file 4:search 5.exit
Enter the choice:2
Name              Sem         Branch
ajay              6           ise
Navya             6           ise

1:write to file 2:display the file 3:modify the file 4:search 5.exit

Enter the choice:4
Enter the record name you want to search:keerthi
Record not found
```

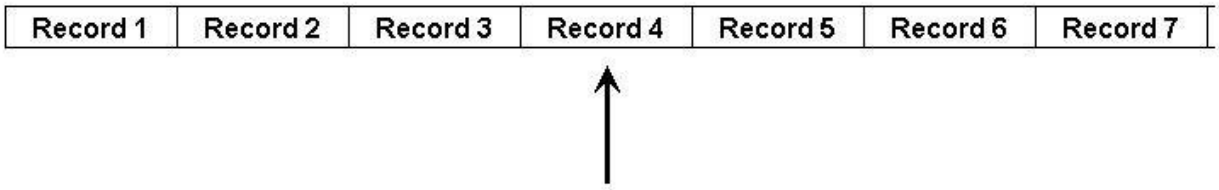
****

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.

RRN(relative record number)

- RRN is an ordinary number that gives the distance of current record from first record. Using RRN, Direct access allows individual records to be read from different locations in the file without reading intervening records.
- When we are using fixed length record, we can calculate the byte offset of each record using the fallowing formula
- ByteOffset = (RRN - 1) × RecLen
    - RRN: relative record number(starts fron 0)
    - RecLen: size of fixed lenth record

Direct Access



PROGRAM 4:

```
#include<iostream.h>
#include<fstream.h>
#include<stdlib.h>
#include<string.h>
#include<stdio.h>
#include <conio.h>
fstream file;
class variable
{
        struct student
        {
                char usn[20],name[20],sem[2],dept[20];
        };
        public: int read_header();
                void write_header(char *);
                void pack();
                void unpack();
                void search();
```

```
};

void variable::write_header(char pos[])
{
        file.open("stu.txt",ios::in|ios::out);
        for(int i=strlen(pos);i!=10;i++)
                pos[i]='%';
        pos[i]='#';
        pos[i+1]='\n';
        file.seekg(0,ios::beg);
        file.write(pos,12);
        file.close();
}

int variable::read_header()
{
        char header[12];
        int count;
        file.open("stu.txt",ios::in);
        file.getline(header,'\n',12);
        file.close();
        count=atoi(header);
        return ++count;
}
void variable::pack()
{
        int len1;
        char b[100],c[150],len[12],temp[12];
        student s;
        cout<<"\nEnter usn, name, sem, dept: ";
        cin>>s.usn>>s.name>>s.sem>>s.dept;
        sprintf(b,"%s|%s|%s|%s|\n",s.usn,s.name,s.sem,s.dept);
        len1=read_header();
        sprintf(c,"%d$",len1);
        strcat(c,b);
        file.open("stu.txt",ios::app);
        file.write(c,strlen(c));
        file.close();
        itoa(len1,len,10);
        write_header(len);
}
void variable::search()
{
        char b[100],rrn[11],temp[100],len[10];
        int flag=0,len1;
        student s;
        file.open("stu.txt",ios::in);
```

```
        cout<<"\nEnter rrn to be searched: ";
        cin>>rrn;
        file.seekg(13,ios::beg);
        while(1)
        {
                if(file.eof())
                        break;
                file.getline(b,10,'$');
                if(strcmp(b,rrn)==0)
                {
                        flag=1;
                        file.getline(b,50,'\n');
                        sscanf(b,"%[^|]|%[^|]|%[^|]|%[^|]",s.usn,s.name,s.sem,s.dept);
                        cout<<s.usn<<" "<<s.name<<" "<<s.sem<<" "<<s.dept<<endl;
                        break;
                }
            file.getline(b,50,'\n');
        }
        if(flag==0)
                cout<<"\nKey not found";
        file.close();
}


void main()
{
        variable f;
        char buffer[10];
        int ch;
        file.open("stu.txt",ios::in|ios::out);
        file.read(buffer,10);
        if(file.eof())
        {
        file.close();
        itoa(-1,buffer,10);
        f.write_header(buffer);
        }
        else
        file.close();
        for(;;)
        {
                cout<<"\n1: Write Record 2: search \nEnter your choice: ";
                cin>>ch;
                switch(ch)
                {
                        case 1:  f.pack();break;
                        case 2: f.search(); break;
```

```
                default: exit(0);
            }
            getch();
            clrscr();
        }
}
```

OUTPUT:

```
1.Insert
2.Search
3.Exit
Enetr your choice:1
Enter the no. of students:2
name = ajay
sem = 6
branch = ise

name = rahul
sem = 6
branch = cse

1.Insert
2.Search
3.Exit
Enetr your choice:2
Enter the RRN to search:1

Record found and details are:"<<
rahul             6           cse
1.Insert
2.Search
3.Exit
Enetr your choice:2
Enter the RRN to search:5
Record not found
```

****

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

Index

A structure containing a set of entries, each consisting of a key field and a reference field, Which is used to locate records in a data file.

Key field

The part of an index which contains keys.

Reference field

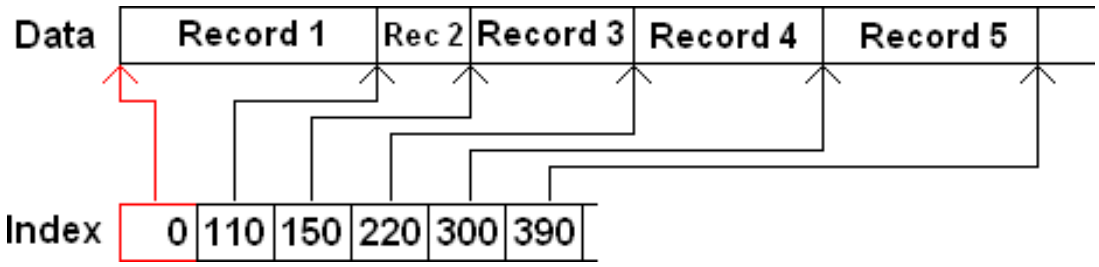
The part of an index which contains information to locate records.

- An index imposes order on a file without rearranging the file.
- Indexing works by indirection.

Simple Index for Entry-Sequenced Files

Simple index

- An index in which the entries are a key ordered linear list. Simple indexing can be useful when theentire index can be held in memory. Changes (additions and deletions) require both the index and the data file to be changed.
- Updates affect the index if the key field is changed, or if the record is moved. An update whichmoves a record can be handled as a deletion followed by an addition.



PROGRAM 5:

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

fstream data,index;
char pri[125][15];
```

```
int ind[125],count=0;
void sort1();
int i,j;

class STUDENT
{
   private:
          char Regno[10], Name[25], Address[50], Sem[5], Branch[10], College[15];
          friend TEXTINDEX;
};
class TEXTINDEX
{
                private:
                        STUDENT a;
                public:
                        void Insert();
                        void Delete();
                        void DisplayI(char *a); /* To display individual record */
                        void Display(STUDENT s);
                        void LoadIndex();
                        void WriteIndex();
};
void TEXTINDEX::Insert()
{
                TEXTINDEX I;
                char Buffer[100],offset[10];
                int pos, high = count;
                data.open("ssk.dat",ios::app);
                data.seekg(0,ios::end);
                pos=data.tellg();
                clrscr();
                cout<<"ENTER RECORD DETAILS\n";
                cout<<"RegNo     : ";
                cin>>a.Regno;
                for(int low=0;low<=high;)
                {
                        int mid = (low+high)/2;
                        int k = atoi(pri[mid]), m = atoi(a.Regno);
                        if(k == m)
                        {
                                cout<<setw(44)<<"DUPLICATE RECORD !!!";
                                return;
                        }
                        else if(m < k)
                                high = mid-1;
                        else if(m > k)
                                low = mid + 1;
```

```
                }
                cout<<"Name     : "; cin>>a.Name;
                cout<<"Address  : "; cin>>a.Address;
                cout<<"Sem      : "; cin>>a.Sem;
                cout<<"Branch   : "; cin>>a.Branch;
                cout<<"College  : "; cin>>a.College;

        sprintf(Buffer,"|%s|%s|%s|%s|%s|%s|#",a.Regno,a.Name,a.Address,a.Sem,a.Branch,a.College);
                data<<Buffer;
                strcpy(pri[count],a.Regno);
                ind[count]=pos;
                count++;
                data.close();
                I.WriteIndex();
}
void TEXTINDEX::Delete()
{
                TEXTINDEX I;
                char reg[20],Buffer[100],usn[15];
                int high = count,flag=0;
                cout<<endl<<"ENTER URN: ";
                cin>>reg;
                data.open("ssk.dat",ios::in | ios::out);
                for(int low=0;low<=high;)
                {
                        int mid = (low+high)/2;
                        int k = atoi(pri[mid]), m = atoi(reg);


                        if(k == m)
                        {
                                flag=1;
                                data.seekg(ind[mid],ios::beg);
                                data.getline(Buffer,100,'#');
                                Buffer[0]='*';
                                pri[mid][0]='*';
                                data.seekg(ind[mid],ios::beg);
                                data<<Buffer;
                                cout<<endl<<"RECORD DELETED";
                                I.WriteIndex();
                                getch();
                                return;
                        }
                        else if(m < k)
                                high = mid-1;
                        else if(m > k)
                                low = mid + 1;
```

```
		}
		if(flag==0)
		{
			cout<<endl<<"RECORD NOT FOUND !!!";
			getch();
		}
		data.close();
}
void TEXTINDEX::DisplayI(char *reg)
{
		int high = count,flag1=0;
		char Buffer[100];
		data.open("ssk.dat",ios::in);
		for(int low=0;low<=high;)
		{
			int mid = (low+high)/2;
			int k = atoi(pri[mid]), m = atoi(reg);
			if(k == m)
			{
				flag1 = 1;
				data.seekg(ind[mid]);
				data.getline(Buffer,100,'#');
	sscanf(Buffer,"|%[^|]|%[^|]|%[^|]|%[^|]|%[^|]|%[^|]#",a.Regno,a.Name,a.Address,a.Sem,a.Branch,a.C
ollege);
				cout<<endl<<"RECORD DETAILS"<<endl;
				cout<<endl<<"URN          : "<<a.Regno;
				cout<<endl<<"NAME         : "<<a.Name;
				cout<<endl<<"ADDRESS      : "<<a.Address;
				cout<<endl<<"SEMESTER     : "<<a.Sem;
				cout<<endl<<"BRANCH       : "<<a.Branch;
				cout<<endl<<"COLLEGE      : "<<a.College<<"\n";
				data.close();
				return;
			}
			else if(m < k)
				high = mid-1;
			else if(m > k)
				low = mid + 1;
		}
			if(flag1==0)
				cout<<endl<<"RECORD DOES NOT EXISTS\n";
			data.close();
}
void TEXTINDEX::LoadIndex()
{
		char buffer[100],temp[100];
		count=0;
```

```
		index.open("index.dat", ios::in);
		while(index)
		{
			index.getline(buffer,100,'#');
			if(index.eof())
			break;
			sscanf(buffer,"|%[^|]|%[^|]",pri[count],temp);
			ind[count]=atoi(temp);
			count++;
		}
		index.close();
}
void TEXTINDEX::WriteIndex()
{
		char buffer[100];
		index.open("index.dat", ios::out);
		sort1();
		for(i=0;i<count;i++)
		{
			itoa(ind[i],buffer,10);
			index<<"|"<<pri[i]<<"|"<<buffer<<"|"<<"#";
		}
		index.close();
}
void main()
{
		TEXTINDEX I;
		char regno[20],sem[20],name[20];
		clrscr();
		I.LoadIndex();
		int true=1,ch;
		while(true)
		{
			int c;
			cout<<"1.INSERT \n"<<"2.DELETE \n"<<"3.SEARCH\n";
			cout<<" ENTER YOUR CHOICE :";
			cin>>c;
			switch(c)
			{
				case 1:I.Insert();
				    break;
				case 2:I.Delete();
				    break;
				case 3:cout<<endl<<"ENTER URN: ";
				    cin>>regno;
				    I.DisplayI(regno);
				    break;
```

```
                        default: exit(0);
                }
                getch();
                clrscr();
        }
}
void sort1()
{
        char temp[20];
        int tempind,k,l;
        for(i=0;i<count;i++)
        {
                for(j=i+1;j<count;j++)
                {
                        k = atoi(pri[i]);
                        l = atoi(pri[j]);
                        if(k > l)
                        {
                                strcpy(temp,pri[i]);
                                strcpy(pri[i],pri[j]);
                                strcpy(pri[j],temp);
                                tempind=ind[j];
                                ind[j]=ind[i];
                                ind[i]=tempind;
                        }
            }
        } }
```

OUTPUT:

```
 1.Insert 2.Display 3.Search 4.Delete 5.Exit
 Enter u'r choice : 1

 Enter the no. of students :2
 Enter the details:
 Name: ajay
 USN: 1vk07is002
 Sem: 6
 Branch: ise
Name: rahul

USN: 1vk07cs045
 Sem: 6
 Branch: cse

 1.Insert 2.Display 3.Search 4.Delete 5.Exit
```

```
Enter u'r choice: 2
ajay        1vk07is002        6           ise
rahul       1vk07cs045        6           cse


1. Insert 2.Display 3.Search 4.Delete 5.Exit
Enter u'r choice :3
Enter USN to search:
1vk07is002
ajay              1vk07is002        6           ise

1. Insert 2.Display 3.Search 4.Delete 5.Exit
Enter u'r choice: 4

Enter USN whose record is to be deleted: 1vk07cs045
Deletion succesfull

1. Insert 2.Display 3.Search 4.Delete 5.Exit
Enter u'r choice: 5
```

****

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<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<stdio.h>
#include<ctype.h>
#include<string.h>
#include<fstream.h>
fstream data,index,secind;
//primary record represetation-->pri[i]|ind[i]
//seconary record representation-->sec[i]|usn[i]
//No of primary records-->count
//No of secondary records-->count1

char pri[125][15],sec[125][40],usn[125][20];
int ind[125],count,count1,i,j;
void sort1(), sort2();

class STUDENT
{
    private:
          char Regno[10], Name[25], Address[50], Sem[5], Branch[10], College[15];
          friend TEXTINDEX;
};
class TEXTINDEX
{
        private:
                STUDENT a;
        public:
                void Insert();
                void Delete();
                void DisplayI(); /* To display individual record */
                void LoadIndex();
                void WriteIndex();
};

void TEXTINDEX::Insert()
{
        TEXTINDEX I;
        fstream data;
        char Buffer[100];
        int high = count, pos;
        data.open("stu.txt",ios::app);
        data.seekg(0,ios::end);
```

```
        pos=data.tellg();
        clrscr();
        cout<<"ENTER RECORD DETAILS\n";
        cout<<"RegNo     : ";
        cin>>a.Regno;
        for(int low=0;low<=high;)
        {
                int mid = (low+high)/2;
                int k = atoi(pri[mid]), m = atoi(a.Regno);
                if(k == m)
                {
                        cout<<"DUPLICATE RECORD !!!";
                        return;
                }
                else if(m < k)
                        high = mid-1;
                else if(m > k)
                        low = mid + 1;
        }
        cout<<"Name     : "; cin>>a.Name;
        cout<<"Address  : "; cin>>a.Address;
        cout<<"Sem      : "; cin>>a.Sem;
        cout<<"Branch   : "; cin>>a.Branch;
        cout<<"College  : "; cin>>a.College;
        sprintf(Buffer,"|%s|%s|%s|%s|%s|%s|#",a.Regno,a.Name,a.Address,a.Sem,a.Branch,     a.College);
        data<<Buffer;
        strcpy(pri[count],a.Regno);//pri record
        ind[count]=pos;
        strcpy(sec[count1],a.Name);//sec record
        strcpy(usn[count1],a.Regno);
        count++;
        count1++;
        I.WriteIndex();
        data.close();
}

int bsearch(int *mid1,int *mid)
{       int low=0, high=count, low1=0, high1=count1, k, m;
        char name[20];
        cout<<"Enter name \n";
        cin>>name;
        for(low=0;low<=high;)
        {
                *mid = (low+high)/2;
                if(strcmpi(sec[*mid],name) == 0)
                {
                        for(low1=0;low1<=high1;)
```

```
                {
                        *mid1 = (low1+high1)/2;
                        k = atoi(pri[*mid1]), m= atoi(usn[*mid]);
                        if(k == m)
                        {
                                return 1;
                        }
                        else if(m < k)
                                        high1 = *mid1-1;
                                else if(m > k)
                                        low1 = *mid1 + 1;
                 }
        }
                else if(strcmpi(sec[*mid],name) > 0)
                        high = *mid-1;
                else if(strcmpi(sec[*mid],name) < 0)
                        low = *mid + 1;
    }
    return 0;
}

void TEXTINDEX::Delete()
{
    TEXTINDEX I;
    char Buffer[100];
    int flag=0,mid,mid1;
    data.open("stu.txt",ios::in | ios::out);
    flag = bsearch(&mid1,&mid);
    if(flag==0)
        cout<<endl<<"RECORD DOES NOT EXISTS\n";
    else
    {
        data.seekg(ind[mid1],ios::beg);
        data.getline(Buffer,100,'#');
        Buffer[0]='*';//delete stu rec
        pri[mid1][0]='*';//delete pri index
        sec[mid][0]='*';//delete sec index
        data.seekg(ind[mid1],ios::beg);
        data<<Buffer;//write deleted record
        cout<<endl<<"RECORD DELETED";
        I.WriteIndex();
        getch();
     }
     data.close();
}

void TEXTINDEX::DisplayI()
```

```
{
        TEXTINDEX I;
        int mid1, mid,flag1=0;
        char Buffer[100],name[20];
        data.open("stu.txt",ios::in|ios::out);
        flag1 = bsearch(&mid1,&mid);
        if(flag1==0)
                cout<<endl<<"RECORD DOES NOT EXISTS\n";
        else
        {
                cout<<"record found \n";
                data.seekg(ind[mid1]);
                data.getline(Buffer,100,'#');

        sscanf(Buffer,"|%[^|]|%[^|]|%[^|]|%[^|]|%[^|]|%[^|]#",a.Regno,a.Name,a.Address,a.Sem,a.Branch,a.C
ollege);
                fflush(stdout);
                cout<<"RECORD DETAILS";
                cout<<endl<<"URN          : "<<a.Regno;
                cout<<endl<<"NAME         : "<<a.Name;
                cout<<endl<<"ADDRESS      : "<<a.Address;
                cout<<endl<<"SEMESTER     : "<<a.Sem;
                cout<<endl<<"BRANCH       : "<<a.Branch;
                cout<<endl<<"COLLEGE      : "<<a.College;
        }
        data.close();
        return;
}

void TEXTINDEX::LoadIndex()
{
        char buffer[100],temp[100];
        count=0;
        index.open("index.txt", ios::in);
        secind.open("sec.txt",ios::in);
        while(index)
        {
                index.getline(buffer,100,'#');
                if(index.eof())
                        break;
                sscanf(buffer,"|%[^|]|%[^|]",pri[count],temp);
                ind[count]=atoi(temp);
                count++;
        }
        count1=0;
        while(secind)
        {
```

```
                secind.getline(buffer,100,'#');
                if(secind.eof())
                        break;
                sscanf(buffer,"|%[^|]|%[^|]",sec[count1],usn[count1]);
                count1++;
        }
        index.close();
        secind.close();
}
void TEXTINDEX::WriteIndex()
{
        char buffer[100];
        index.open("index.txt", ios::out);
        secind.open("sec.txt",ios::out);
        sort1();
        sort2();
        for(i=0;i<count;i++)
        {
                itoa(ind[i],buffer,10);
                index<<"|"<<pri[i]<<"|"<<buffer<<"|"<<"#";//pri record writing
        }
        for(i=0;i<count1;i++)
        {
                secind<<"|"<<sec[i]<<"|"<<usn[i]<<"|"<<"#";//sec record writing
        }
        index.close();
        secind.close();
}

void main()
{
        TEXTINDEX I;
        char name[20];
        clrscr();
        I.LoadIndex();
        int true=1,ch;

        while(true)
        {
                int c;
                cout<<"   1->INSERT RECORD\n";
                cout<<"   2->DELETE RECORD\n";
                cout<<"   3->DISPLAY INDIVIDUAL RECORD\n";
                cout<<"   4->QUIT\n";
                cout<<" ENTER YOUR CHOICE :";
                cin>>c;
                switch(c)
```

```
                {
                        case 1:I.Insert();
                             break;
                        case 2:I.Delete();
                             break;
                        case 3:I.DisplayI();
                             break;
                        default: exit(0);
                }
                getch();
                clrscr();
        }
}

void sort1()
{
        char temp[20];
        int tempind;
        for(i=0;i<count;i++)
        {
                for(j=i+1;j<count;j++)
                {
                     if(strcmpi(pri[i],pri[j])>0)
                     {
                                strcpy(temp,pri[i]);
                                strcpy(pri[i],pri[j]);
                                strcpy(pri[j],temp);
                                tempind=ind[j];
                                ind[j]=ind[i];
                                ind[i]=tempind;
                        }
                }
        }
}

void sort2()
{
        char temp[40];
        for(i=0;i<count;i++)
        {
                for(j=i+1;j<count;j++)
                {
                        if(strcmp(sec[j],sec[i])<0)
                        {
                                strcpy(temp,sec[i]);
                                strcpy(sec[i],sec[j]);
                                strcpy(sec[j],temp);
```

```
                    strcpy(temp,usn[i]);
                    strcpy(usn[i],usn[j]);
                    strcpy(usn[j],temp);
                }
            }
        }}
```

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

Cosequential operations

Operations which involve accessing two or more input files sequentially and in parallel, resulting in one or more output files produced by the combination of the input data.

Considerations for Cosequential Algorithms

- Initialization - What has to be set up for the main loop to work correctly?
- Getting the next item on each list - This should be simple and easy, from the main algorithm.
- Synchronization - Progress of access in the lists should be coordinated.
- Handling End-Of-File conditions - For a match, processing can stop when the end of any list is reached.
- Recognizing Errors - Items out of sequence can "break" the synchronization.

Matching Names in Two Lists

Match

The process of forming a list containing all items common to two or more lists.

Cosequential Match Algorithm

- Initialize (open the input and output files.)
- Get the first item from each list.
- While there is more to do:

    Compare the current items from each list.

        If the items are equal,

            Process the item.

Get the next item from each list.

Set *more* to true iff none of this lists is at end of file.

If the item from list *A* is less than the item from list *B*,

Get the next item from list *A*.

Set *more* to true iff list *A* is not at end-of-file.

If the item from list *A* is more than the item from list *B*,

Get the next item from list *B*.

Set *more* to true iff list *B* is not at end-of-file.

Finalize (close the files.)

PROGRAM 7:

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<fstream.h>
#include<stdio.h>
fstream list[3];
class intersect
{
        public:
                void sort(char [25][30],int);
                void input(char [25][30],int);
};
void intersect::sort(char s[25][30],int count)
{
        int i,j;
        char temp[40];
        for(i=0;i<count;i++)
                for(j=i+1;j<count;j++)
                        if(strcmp(s[i],s[j])>0)
                        {
                                strcpy(temp,s[i]);
                                strcpy(s[i],s[j]);
                                strcpy(s[j],temp);
                        }
}
void intersect::input(char str[25][30],int lnum)
{
        if(lnum == 0)
                list[0].open("list0.txt",ios::out);
        else
                list[1].open("list1.txt",ios::out);
        int i,j;
        for(i=0;;i++)
        {
                gets(str[i]);
```

```
            if(strcmp(str[i],"#")==0)
                  break;
      }
      sort(str,i);



      for(j=0;j<i;j++)
      {
            list[lnum]<<str[j]<<"\n";
      }
      list[0].close();
      list[1].close();
}
void main()
{
      char str0[25][30],str1[25][30];
      char s0[30],s1[30];
      int i,j;
      intersect I;
      clrscr();
      cout<<"Enter the names of list1(To terminate enter string #)\n";
      I.input(str0,0);
      cout<<"Enter the names of list2(To terminate enter string #)\n";
      I.input(str1,1);
      list[0].open("list0.txt",ios::in);
      list[1].open("list1.txt",ios::in);
      list[2].open("list2.txt",ios::out);
      list[0].getline(s0,50,'\n');
      list[1].getline(s1,50,'\n');
      while(!list[0].eof() && !list[1].eof())
      {
            if(strcmp(s0,s1)==0)
            {
                  list[2]<<s0<<endl;
                  list[0].getline(s0,50,'\n');
                  list[1].getline(s1,50,'\n');
            }
            else if(strcmp(s0,s1)<0)
                  list[0].getline(s0,50,'\n');
            else
                  list[1].getline(s1,50,'\n');
      }
      if(strcmp(s0,s1)==0)
            cout<<s0<<endl;
      list[0].close();
      list[1].close();
```

```
        list[2].close();
        list[2].open("list2.txt",ios::in);
        list[2].getline(s1,50,'\n');
        if(strcmp(s1,"\0")==0)
                cout<<" No matching strings found:"<<endl;
        else
        {
                cout<<"Names in both the lists are"<<endl;
                while(1)
                {
                        cout<<s1<<endl;
                        list[2].getline(s1,50,'\n');
                        if(list[2].eof())
                                break;
                }
        }
        getch();
}
```

****

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

Merge

The process of forming a list containing all items in any of two or more lists.

K-way merge

A merge of order k.

Order of a merge

The number of input lists being merged.

- If the distribution phase creates *k* runs, a single k-way merge can be used to produce the final sorted file.
- A significant amount of seeking is used by a k-way merge, assuming the input runs are on the same disk.

PROGRAM-8

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

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

int no;
fstream file[8];
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 recrd[20];int k;
         k=0;
         fstream f1,f2;
         f1.open(file1,ios::in);                    //open the first file
         f2.open(file2,ios::in);                    //open the second file
         while(!f1.eof())                           //Unpack and retrieve first file
         {
```

```
            f1.getline(recrd[k].name,20,'|');
            f1.getline(recrd[k++].usn,20,'\n');
        }
        while(!f2.eof())                                    //Unpack and retrieve second file
        {
            f2.getline(recrd[k].name,20,'|');
            f2.getline(recrd[k++].usn,20,'\n');
        }
        record temp;
        int t,y;
        for(t=0;t<k-2;t++)                                  //Sort the retrieved records
        for(y=0;y<k-t-2;y++)
        if(strcmp(recrd[y].name,recrd[y+1].name)>0)
        {
            temp=recrd[y];
            recrd[y]=recrd[y+1];
            recrd[y+1]=temp;
        }
        fstream temp1;
        temp1.open(filename,ios::out);                      //Open the file to be packed into
        for(t=1;t<k-1;t++)                                  //Pack the sorted records onto the file
        temp1<<recrd[t].name<<"|"<<recrd[t].usn<<"\n";
        f1.close();f2.close();temp1.close();
        return;
}


void kwaymerge()
{
        char filename[7][20]={"11.txt","22.txt","33.txt","44.txt","111.txt","222.txt","1111.txt"};
        int i;
        int k;
        k=0;
        for(i=0;i<8;i+=2)                                   //Merge and sort the 8 original files onto
        {                                                   //the four files indicated {11.txt,22.txt....}
            merge_file(fname[i],fname[i+1],filename[k++]);
        }
        k=4;
        for(i=0;i<4;i+=2)               //Merge and sort the four files onto 111.txt and 222.txt
        {
            merge_file(filename[i],filename[i+1],filename[k++]);
        }
        merge_file(filename[4],filename[5],filename[6]);    //Merge and sort the two files
        return;                                             // onto the 1111.txt file
}
```

```
int main()
{
        int i;
        cout<<"Enter the no. of records : ";
        cin>>no;
        cout<<"\nEnter the details : \n";
        for(i=0;i<8;i++)                        //Create 8 files to store the split data
        file[i].open(fname[i],ios::out);
        for(i=0;i<no;i++)                       //Split and pack data onto the files
        {
                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();                            //Merge
        fstream result;
        result.open("1111.txt",ios::in);
        cout<<"\nSorted Records : \n";
        char name[20],usn[20];
        for(i=0;i<no;i++)                       //Unpack the sorted records and display
        {
                result.getline(name,20,'|');
                result.getline(usn,20,'\n');
                cout<<"\nName   : "<<name<<"\nUSN    : "<<usn<<"\n";
        }
        return 0;
}
```