FILE STRUCTURES LABORATORY WITH MINI PROJECT

Subject Code	17ISL68	Credits	02
Number of Lecture Hours/Week	1I+ 2P	CIE Marks	40
		SEE Marks	60
Total hours allotted	40	Exam Hours	03

Part-A: Laboratory Experiments

Design, develop, and implement the following programs

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

Part-B: Mini Project

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

Course outcome:

The students should be able to:

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

Conduction of Practical Examination:

1. All laboratory experiments from part A are to be included for practical examination.
2. Mini project has to be evaluated for 40 Marks as per 6(b).
3. Report should be prepared in a standard format prescribed for project work.
4. Students are allowed to pick one experiment from the lot.
5. Strictly follow the instructions as printed on the cover page of answer script.
6. Marks distribution:
    a. Part A: Procedure + Conduction + Viva: 09 + 42 +09 = 60 Marks
    b. Part B: Demonstration + Report + Viva voce = 20+14+06 = 40 Marks
7. Change of experiment is allowed only once, and marks allotted to the procedure part to be made zero.

Books/References:

1. Michael J. Folk, Bill Zoellick, Greg Riccardi, **File Structures - An Object-Oriented Approach with C++**, 3rd Edition, Pearson Education, 1998.
2. K.R. Venugopal, K.G. Srinivas, P.M. Krishnaraj, **File Structures Using C++**, Tata McGraw-Hill, 2008.
3. Scot Robert Ladd, **C++ Components and Algorithms**, BPB Publications, 1993.

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

Algorithm:

Step 1: Start

Step 2: Enter your choice 1. From standard I/O 2. From Files

Step 3: Select the choice- 1: Using Standard file 2: Using Files

Case 1: [using standard I/O]

Enter the number of names to be read

Read the value of N

a. Read one name at a time

b. call reverse(name) function

display the reversed name on standard output

Repeat a and b for N times

Case 2: [Using Files]

Open the file for reading

Read names from an input file

Reverse the name by calling strrev(name) and redirect results to a file.

Step 4: Close the file

Step 5: Stop

Program:

```
// Program to read the series of names, one per line, from input and write these names spelled in reverse order to the output using I/O redirection and pipes
#include<iostream.h>
#include<stdio.h>
#include<string.h>
#include<fstream.h>
#include<conio.h>
#include<iomanip.h>
#include<stdlib.h>
```

```
class std_file
{
        private:
                char name[10][20];
                char input[20], output[20],str[20];
        public:
                void std_io();
                void file_io();
};
void std_file::std_io()
{
        intn,i;
        cout<<"Enter the number of names to read "<<endl;
        cin>>n;
        cout<<"Enter the names"<<endl;
        for(i=0;i<n;i++)
        gets(name[i]);
        cout<<"The reversed names are"<<endl;
        for(i=0;i<n;i++)
        cout<<strrev(name[i])<<endl;
}
void std_file::file_io()
{
        fstreamifile,ofile;
        cout<<"Enter the filename which contain list of names"<<endl;
        cin>>input;
        ifile.open(input,ios::in);
        if(!ifile)
        {
                cout<<"File does not exist";
                exit(0);
                getch();
```

```
        }
        cout<<"Enter the filename to store names in reverse order"<<endl;
        cin>>output;
        ofile.open(output,ios::out);
        while(!ifile.eof())
        {
                ifile.getline(str,20);
                ofile<<strrev(str)<<endl;       //to reverse string characters
        }
}
void main()
{
        int num;
        std_file s;
        clrscr();
        for(;;)
        {
                cout<<"1: Using standard I/O 2: Using files"<< endl;
                cout<<"Enter the choice";
                cin>>num;
                switch(num)
                {
                        case 1: s.std_io();
                                break;
                        case 2: s.file_io();
                                break;
                        default: exit(0);
                }
        }
}
```

Output 1:

1: Using standard I/O 2: Using files

Enter the choice 1

Enter the number of names to read 3

Enter the names

akash

rama

Deepak

The reversed names are

hsaka

amar

kapeed

Viva Voce:

1. What is file structure?

   File structure is a combination of representations for data in files and of operations for accessing the data.

2. Define File.

   A file is a container in a computer system for storing information.

3. List different types of file.

   Types of files are: Physical files and Logical files.

4. What is datatype?

   It is an attribute of data which tells the compiler or interpreter how the programmer intends to use the data.

5. List different types of datatypes.

   Integer, Floating-point number, Character, String, Boolean.

6. Difference between data structure and file structure.

   The representation of the particular data structure in the memory of a computer is called a storage structure whereas a storage structure representation in auxiliary memory is often called a file structure.

7. List basic file input operations used.

   Writing into the file

8. List basic file output operations used.

   Reading from the file

9. Basic file operations and their syntactic structures in C and C++

   In C: fopen(fname, ftype), fread(void *buf, size_t size, size_t num, FILE *fp)

   In C++: Using constructor: ifstream fin; fin.open("filename");

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

Algorithm:

Step 1: Start
Step 2: Enter the choice
1.Write 2. Display 3. Search 4. Modify 5. Exit
case 1: write() //Read the name, usn, age, sem, branch
case 2: unpack()
display()
case 3: unpack()
search()
case 4: unpack()
search()
modify()
Step 3: Stop

Algorithm: write()
Step 1: Start
Step 2: Enter the name, usn, age, sem, branch
Step 3: call pack(t) function
Step 4: Stop

Algorithm: pack(t)
Step 1: Start
Step 2: Open a file hello.txt
Step 3: copy the strings- name, usn, age, branch and concatenate in buffer
Step 4: Now read length of buffer -strlen(buffer)
Step 5: close a file
Step 6: Stop

Algorithm: unpack()
Step 1: Start
Step 2: open a file hello.txt
Step 3: Read each line from the buffer -getline(buffer,100)
Step 5: close a file
Step 6: Stop

Algorithm: display()
Step 1: Start
Step 2: if(count==0), print No records
else read name, usn, age, sem, branch
Step 3: Print name, usn, age, sem, branch
Step 4: Stop

Algorithm: search()
Step 1: Start
Step 2: Enter the usn
Step 3: Compare usn with the buffer
if(i==count)
print as Record not found
else
print the name, usn, age, sem and branch
Step 4: Stop

Algorithm: modify()
Step 1: Start
Step 2: Enter new value to modify
Step 3: Update the name, usn, age, sem, branch
Step 4: call pack(s[j]);
Step 5: Stop

Program:

```
// Program to read and write and student objects with fixed-length records and the fields delimited by "|"
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<string.h>
#include<fstream.h>
class student
{
        public: char name[20], usn[10], age[5], sem[5],branch[5];
};
student s[100], t;
char buffer[45], temp[20];
int count=0, i;
fstream fp;
void pack(student p)
{
        fp.open("hello.txt",ios::app);
        strcpy(buffer,p.name);
        strcat(buffer,"|");
```

```
        strcat(buffer,p.usn);
        strcat(buffer,"|");
        strcat(buffer,p.age);
        strcat(buffer,"|");
        strcat(buffer,p.sem);
        strcat(buffer,"|");
        strcat(buffer,p.branch);
        int x=strlen(buffer);
        for(int j=0;j<45-x;j++)
        strcat(buffer,"!");
        fp<<buffer<<endl;
        fp.close();
}
void write()
{
        cout<<"Enter the name\n";
        cin>>t.name;
        cout<<"Enter the usn\n";
        cin>>t.usn;
        cout<<"Enter the age\n";
        cin>>t.age;
        cout<<"Enter the sem\n";
        cin>>t.sem;
        cout<<"Enter the branch\n";
        cin>>t.branch;
        pack(t);
}
void unpack()
{
        fp.open("hello.txt",ios::in);
        for(i=0;i<count;i++)
        {
```

```
                fp.getline(buffer,100);
                sscanf(buffer,"%[^|]|%[^|]|%[^|]|%[^|]|%[^!]!",
                s[i].name,s[i].usn,s[i].age,s[i].sem,s[i].branch);
        }
        fp.close();
}
void display()
{
        if(count==0)
        {
                cout<<"\nNo records\n";
                return;
        }
        cout<<"\n name\t usn\t age\t sem\t branch\n";
        for(i=0;i<count;i++)
        cout<<s[i].name<<"\t"<<s[i].usn<<"\t"<<s[i].age<<"\t"<<s[i].sem<<"\t"<<s[i].bra
nch<<endl;
}
void search()
{
        cout<<"Enter the usn\n";
        cin>>temp;
        for(i=0;i<count;i++)
        if(!strcmp(s[i].usn,temp))
        {
                cout<<"Record found\n"<<s[i].name<<"\t"<<s[i].usn<<"\t"
                <<s[i].age<<"\t" <<s[i].sem<<"\t"<<s[i].branch<<endl;
                break;
        }
        if(i==count)
        cout<<"Record not found";
}
```

```
void modify()
{
        if(i==count)
        return;
        cout<<"Enter new values\n Enter name :";
        cin>>s[i].name;
        cout<<"Enter usn :";
        cin>>s[i].usn;
        cout<<"Enter age :";
        cin>>s[i].age;
        cout<<"Enter sem :";
        cin>>s[i].sem;
        cout<<"Enter branch :";
        cin>>s[i].branch;
        fp.close();
        remove("hello.txt");
        fp.open("hello.txt",ios::out);
        fp.close();
        for(int j=0;j<count;j++)
        pack(s[j]);
}
void main()
{
        int c;
        clrscr();
        fp.open("hello.txt",ios::out);
        fp.close();
        while(1)
        {
                cout<<"\n1.Write\n 2. Display\n 3. Search\n 4.Modify\n 5.Exit\n;
                cout<< Enter your choice\n";
```

```
            cin>>c;
            switch(c)
            {
                case 1: count++; write();break;
                case 2: unpack(); display();break;
                case 3: unpack(); search();break;
                case 4: unpack(); search();modify();break;
                default:exit(0);
            }
        }
}
```

Output:

```
1. Write        2 .Display      3. Search       4. Modify       5. Exit
Enter your choice: 1
Enter the name : Anusha
Enter the usn: 4su16is001
Enter the age: 21
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .Display      3. Search       4. Modify       5. Exit
Enter your choice: 1
Enter the name : Divya
Enter the usn: 4su16is007
Enter the age: 20
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .Display      3. Search       4. Modify       5. Exit
Enter your choice: 2
Name    usn             age     sem     branch
Anusha  4su16is001      21      6       ISE
Divya   4su16is007      20      6       ISE

1. Write        2 .Display      3. Search       4. Modify       5. Exit
Enter your choice: 3
Enter the USN: 4su16is001
Record found
Name    usn             age     sem     branch
Anusha  4su16is001      21      6       ISE
```

```
1. Write        2 .Display      3. Search       4. Modify       5. Exit
Enter your choice: 4
Enter the USN: 4su16is007
Record found
Enter new values
Enter name: Arpita
Enter USN: 4su16is003
Enter age: 22
Enter sem: 5
Enter branch: CSE

1. Write        2 .Display      3. Search       4. Modify       5. Exit
Enter your choice: 2
Name    usn             age     sem     branch
Arpita  4su16is003      22      5       CSE
Divya   4su16is007      20      6       ISE
1. Write        2 .Display      3. Search       4. Modify       5. Exit
Enter your choice: 5
```

Viva Voce:

1. Define Field.

   Each element of a record is known as field.

2. Define record.

   Set of fields constitutes a file is called record.

3. What do you mean by delimiter? List any two delimiters.

   Special mark used to differentiate between two fields. Ex: comma(,), pipe(|)

4. Differentiate between field and record.

   A field is smaller unit of a record and record is set of fields constitutes a file.

5. What do you mean by fixed length record?

   A fixed-length record is one in which every field has a fixed length.

6. Differentiate fixed length record with variable length record?

   A fixed-length record is one in which every field has a fixed length. A variable-length record has at least one variable-length field.

7. What are the different ways of identifying fields?

   Fixed Length Fields, Delimited Variable Length Fields, Length Prefixed Variable Length Fields, Representing Record or Field Length, Tagged Fields.

8. What are the different ways of identifying record structures?

Fixed Length Records, Delimited Variable Length Records, Length Prefixed Variable Length Records.

9. What do you mean by pack and unpack?

   The pack and unpack are the commands which are primarily used for creating or reading binary structures.

10. How do we delete fixed length records?

    Using file header.

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

Algorithm:

Step 1: Start
Step 2: Enter the choice
1.Write 2. Display 3. Search 4. Modify 5.Exit
case 1: write() //Read the name, usn, age, sem, branch
case 2: unpack()
display()
case 3: unpack()
search()
case 4: unpack()
search()
modify()
Step 3: Stop

Algorithm: write()
Step 1: Start
Step 2: Enter the name, usn, age, sem, branch
Step 3: call pack(t) function
Step 4: Stop

Algorithm: pack(t)
Step 1: Start
Step 2: Open a file hello.txt
Step 3: copy the strings- name, usn, age, branch and concatenate in buffer
Step 4: Now read length of buffer -strlen(buffer)
Step 5: close a file
Step 6: Stop

Algorithm: unpack()
Step 1: Start
Step 2: open a file hello.txt
Step 3: Read each line from the buffer -getline(buffer,100)
Step 5: close a file
Step 6: Stop

Algorithm: display()
Step 1: Start
Step 2: if(count==0), print No records
else read name, usn, age, sem, branch
Step 3: Print name, usn, age, sem, branch
Step 4: Stop

Algorithm: search()
Step 1: Start
Step 2: Enter the usn
Step 3: if(!strcmp(s[i].usn,temp))
print the name, usn, age, sem and branch
else
print as Record not found
Step 4: Stop

Algorithm: modify()
Step 1: Start
Step 2: Enter new value to modify
Step 3: Update the name, usn, age, sem, branch
Step 4: call pack(s[j]);
Step 5: Stop

Program:

// Program to read and write and student objects with variable-length records using any suitable record structure

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<string.h>
#include<fstream.h>
class student
{
        public:char name[20], usn[10], age[5], sem[5],branch[5];
};
student s[100],t;
char buffer[45],temp[20];
int count=0,i;
fstream fp;
void pack(student p)
{
        fp.open("hello.txt",ios::app);
        strcpy(buffer,p.name);
        strcat(buffer,"|");
```

```
        strcat(buffer,p.usn);
        strcat(buffer,"|");
        strcat(buffer,p.age);
        strcat(buffer,"|");
        strcat(buffer,p.sem);
        strcat(buffer,"|");
        strcat(buffer,p.branch);
        strcat(buffer,"|");
        fp<<buffer<<endl;
        fp.close();
}
void write()
{
        cout<<"Enter the name\n";
        cin>>t.name;
        cout<<"Enter the usn\n";
        cin>>t.usn;
        cout<<"Enter the age\n";
        cin>>t.age;
        cout<<"Enter the sem\n";
        cin>>t.sem;
        cout<<"Enter the branch\n";
        cin>>t.branch;
        pack(t);
}
void unpack()
{
        fp.open("hello.txt",ios::in);
        for(i=0;i<count;i++)
        {
                fp.getline(buffer,100);
```

```
                sscanf(buffer,"%[^|]|%[^|]|%[^|]|%[^|]|%[^|]|",s[i].name,s[i].usn,
                s[i].age,s[i].sem,s[i].branch);
        }
        fp.close();
}
void display()
{
        if(count==0)
        {
                cout<<"\nNo records\n";
                return;
        }
        cout<<"\n name\t usn\t age\t sem\t branch\n";
        for(i=0;i<count;i++)
        cout<<s[i].name<<"\t"<<s[i].usn<<"\t"<<s[i].age<<"\t"<<s[i].sem<<
        "\t"<<s[i].branch<<endl;
}
void search()
{
        cout<<"Enter the usn\n";
        cin>>temp;
        for(i=0;i<count;i++)
        if(!strcmp(s[i].usn,temp))
        {
                cout<<"Record found\n"<<s[i].name<<"\t"<<
                s[i].usn<<"\t"<<s[i].age<<"\t"<<s[i].sem<<"\t"<<s[i].branch<<endl;
                break;
        }
        if(i==count)
        cout<<"Record not found";
}

void modify()
```

```
{
        if(i==count)
        return;
        cout<<"Enter new values\n Enter name :";
        cin>>s[i].name;
        cout<<"Enter usn :";
        cin>>s[i].usn;
        cout<<"Enter age :";
        cin>>s[i].age;
        cout<<"Enter sem :";
        cin>>s[i].sem;
        cout<<"Enter branch :";
        cin>>s[i].branch;
        fp.close();
        remove("hello.txt");
        fp.open("hello.txt",ios::out);
        fp.close();
        for(int j=0;j<count;j++)
        pack(s[j]);
}
void main()
{
        int c;
        clrscr();
        fp.open("hello.txt",ios::out);
        fp.close();
        while(1)
        {
                cout<<"\n1.Write\n 2.Display\n 3.Search\n 4.Modify\n 5.Exit\n Enter your
choice\n";
                cin>>c;
                switch(c)
```

```
        {
                case 1:count++;write();break;
                case 2:unpack();display();break;
                case 3:unpack();search();break;
                case 4:unpack();search();modify();break;
                default:exit(0);
        }
    }
}
```

Output:

```
1. Write        2 .Display      3. Search       4. Modify       5. Exit
Enter your choice: 1
Enter the name : Arun
Enter the usn: 4su16is002
Enter the age: 20
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .Display      3. Search       4. Modify       5. Exit
Enter your choice: 1
Enter the name: Kiarn
Enter the usn: 4su16is013
Enter the age: 21
Enter the Sem: 6
Enter the branch: ISE

1. Write        2 .Display      3. Search       4. Modify       5. Exit
Enter your choice: 2
Name    usn             age     sem     branch
Arun    4su16is002      20      6       ISE
Kiran   4su16is013      21      6       ISE

1. Write        2 .Display      3. Search       4. Modify       5. Exit
Enter your choice: 2
Enter the USN: 4su16is013
Record found
Name    usn             age     sem     branch
Kiran   4su16is013      21      6       ISE

1. Write        2 .Display      3. Search       4. Modify       5. Exit
```

```
Enter your choice: 4
Enter the USN: 4su16is007
Record found
Name    usn          age   sem   branch
Kiran   4su16is013   21    6     ISE
Enter new values
Enter name: Harsha
Enter USN:    4su16is009
Enter age: 21
Enter sem: 6
Enter branch: CSE

1. Write    2 .Display    3. Search    4. Modify    5. Exit
Enter your choice: 2
Name    usn          age   sem   branch
Arun    4su16is002   20    6     ISE
Harsha  4su16is009   21    6     CSE

1. Write    2 .Display    3. Search    4. Modify    5. Exit
Enter your choice: 5
```

Viva Voce:

1. What do you mean by a record?

   Record is the set of fields that constitutes a file.
2. How do you represent a record in a file?

   Using fields.
3. Explain need for variable length records.

   Conserves disk space by using just the space needed to hold variable length data.
4. Explain different ways of representing variable length records.

   Storage of multiple record types in a file, record types that allow variable lengths for one or more fields, record types that allow repeating fields
5. Differentiate between fixed and variable length records.

   Fixed-length record is one in which every field has a fixed length. A variable-length record has at least one variable-length field.
6. What are the different ways of identifying fields?

   Fixed Length Fields, Delimited Variable Length Fields, Length Prefixed Variable Length Fields, Representing Record or Field Length, Tagged Fields.
7. What are the different ways of identifying record structures.?

Fixed Length Records, Delimited Variable Length Records, Length Prefixed Variable Length Records.

8. What do you mean by pack and unpack?

   The pack and unpack are the commands which are primarily used for creating or reading binary structures.

9. How do we delete variable length records?

   Using Avail List.

10. Define sequential access.

    Access to a file that requires the user to read through the file from the beginning in the order in which it is stored.

4. Write a program to write student objects with variable-length records using any suitable record structure and to read from this file a student record using RRN.

Algorithm:

Step 1: Start
Step 2: Enter the choice
1.Write 2. Search 3.Exit
case 1: write() //Read the name, usn, age, sem, branch
case 2: search()
Step 3: Stop

Algorithm: write()
Step 1: Start
Step 2: Enter the name, usn, age, sem, branch
Step 3: call pack(t) function
Step 4: Stop

Algorithm: pack(t)
Step 1: Start
Step 2: Open a file hello.txt
Step 3: copy the strings- name, usn, age, branch and concatenate in buffer
Step 4: Now read length of buffer -strlen(buffer)
Step 5: close a file
Step 6: Stop

Algorithm: unpack()
Step 1: Start
Step 2: open a file hello.txt
Step 3: Read each line from the buffer -getline(buffer,100)
Step 5: close a file
Step 6: Stop

Algorithm: search()
Step 1: Start
Step 2: Enter the rrn
Step 3: if(strcmp(temp1,temp2)==0)
print the name, usn, age, sem and branch
else
print as Record not found
else
print the name, usn, age, sem and branch
Step 4: Stop

Program:

```
// Program to write student objects with variable-length records using any RRN
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<string.h>
#include<fstream.h>
class student
{
        public:char name[20],usn[10],age[5],sem[5],branch[5];
};

student s[100],t;
char buffer[45],temp1[20],temp2[20];
int count=0,i;
fstream fp;
void pack(student p)
{
        fp.open("hello.txt",ios::app);
        strcpy(buffer,p.name);
        strcat(buffer,"|");
        strcat(buffer,p.usn);
        strcat(buffer,"|");
        strcat(buffer,p.age);
        strcat(buffer,"|");
        strcat(buffer,p.sem);
        strcat(buffer,"|");
        strcat(buffer,p.branch);
        strcat(buffer,"|");
    //  int x=strlen(buffer);
     //  for(int j=0;j<45-x;j++)
     //  strcat(buffer,"!");
```

```
        fp<<count<<"|"<<buffer<<endl;
        fp.close();
}
void write()
{
        cout<<"Enter the name\n";
        cin>>t.name;
        cout<<"Enter the usn\n";
        cin>>t.usn;
        cout<<"Enter the age\n";
        cin>>t.age;
        cout<<"Enter the sem\n";
        cin>>t.sem;
        cout<<"Enter the branch\n";
        cin>>t.branch;
        pack(t);
}
void search()
{
        cout<<"Enter the rrn\n";
        cin>>temp1;
        fp.open("hello.txt",ios::in);
        for(i=0;i<count;i++)
        {
                fp.getline(buffer,100);
                sscanf(buffer,"%[^|]|%[^|]|%[^|]|%[^|]|%[^|]|%[^|]|",temp2,s[i].name,
                s[i].usn,s[i].age,s[i].sem,s[i].branch);
                if(strcmp(temp1,temp2)==0)
                {
                        cout<<"Record found\n"<<s[i].name<<"\t"<<s[i].usn<<"\t"<<
                        s[i].age<<"\t" <<s[i].sem<<"\t"<<s[i].branch<<endl;
                        break;
                }
```

```
        }
        if(i==count)
        cout<<"Record not found";
        fp.close();
}
void main()
{
        int c;
        clrscr();
        fp.open("hello.txt",ios::out);
        fp.close();
        while(1)
        {
                cout<<"\n1.Write\n 2. Search \n 3.Exit\n Enter your choice\n";
                cin>>c;
                switch(c)
                {
                        case 1:count++;write();break;
                        case 2:search();break;
                        default:exit(0);
                }
        }
}
```

Output:

```
1. Write        2 .search       3. Exit
Enter your choice: 1
Enter the name : Sandesh
Enter the usn: 4su16IS020
Enter the age: 22
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .search       3. Exit
```

```
Enter your choice: 1
Enter the name : kiran
Enter the usn: 4su16is013
Enter the age: 21
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .search       3. Exit
Enter your choice: 1
Enter the name : Namratha
Enter the usn: 4su16is012
Enter the age: 20
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .search       3. Exit
Enter your choice: 2
Enter the rrn :2
Record found
Name    usn             age     sem     branch
Kiran   4su16is013      21      6       ISE

1. Write        2 .search       3. Exit
Enter your choice: 2
Enter the rrn :4
Record not found
1. Write        2 .search       3. Exit
Enter your choice: 3
```

Viva Voce:

1. Define RRN.

   Relative Record Number identifies which position in a file the record is in.

2. What are the different modes of opening a file?

   Read, write and append modes.

3. How to close a file? What happens if a file is not closed?

   Using fclose() function. Run out of file descriptors/handles available for a process and attempting to open more files will fail eventually.

4. What is the difference between read() and getline()?

   getline() reads a single whole line. read() reads a block of data.

5. List different types of record structures

Variable length records and Fixed length records.

6. Explain how RRN is used to retrieve fixed length record.

   RRN is used to calculate the byte offset of the start of the record relative to the start of the file.

7. Explain how RRN is used to retrieve variable length record.

   To get the record required read sequentially through the file by counting records.

8. List the string operation performed on a list of records.

   String operations- strrev, strlen, strcat, strcmp

9. What is Fragmentation?

   Fragmentation refers to the condition of a disk in which files are divided into pieces scattered around the disk.

10. Differentiate between Logical and Physical file

    Physical files contain the actual data that is stored on the system. Logical files do not contain data. They contain a description of records found in one or more physical files.

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

Algorithm:

Step 1: Start
Step 2: Open file std.txt and ind.txt
Step 3: Enter the choice
1.Write 2. Search 3. Delete 4. Exit
case 1: write() //Read the name, usn, age, sem, branch
case 2: search()
case 3: search()
delet()
Step 3: Stop

Algorithm: write()
Step 1: Start
Step 2: Enter the name, usn, age, sem, branch
Step 3: call pack(t) function
Step 4: Stop

Algorithm: pack(t)
Step 1: Start
Step 2: Open a file hello.txt
Step 3: copy the strings- name, usn, age, branch and concatenate in buffer
Step 4: Now read length of buffer -strlen(buffer)
Step 5: close a file
Step 6: Stop

Algorithm: unpack()
Step 1: Start
Step 2: open a file hello.txt
Step 3: Read each line from the buffer -getline(buffer,100)
Step 5: close a file
Step 6: Stop

Algorithm: search()
Step 1: Start
Step 2: Enter the USN to search for
Step 3: if(strcmp(tempusn,keyusn)==0)
print Record found";
rrn=atoi(temprrn);
Calculate RRN value of record by seekg((rrn-1)*49
print temprrn, name, usn, age, sem, branch
else
print as Record not found
Step 4: close file pointe fp1 and fp2

Step 5: Stop

Algorithm: delet()
Step 1: Start
Step 2: open a file std.txt
Step 3: scan the file and store the name, usn, age, sem, branch into buffer
Step 4: close a file
Step 5: remove the record from std.txt and ind.txt
Step 6: Stop

Program:

```
// Program to implement simple index on primary key for a file of student objects
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<string.h>
#include<fstream.h>
#include<stdlib.h>
class student
{
        public:char name[20],usn[10],age[5],sem[5],branch[5];
};
student s[100],t;
char buffer[50],buffer1[50];
char tempusn[15],temprrn[10],keyusn[15];
int count=0,i;
fstream fp1,fp2;
void pack(student p)
{
        fp1.open("std.txt",ios::app);
        fp2.open("ind.txt",ios::app);
        strcpy(buffer,p.name);
        strcat(buffer,"|");
        strcat(buffer,p.usn);
        strcat(buffer,"|");
```

```
        strcat(buffer,p.age);
        strcat(buffer,"|");
        strcat(buffer,p.sem);
        strcat(buffer,"|");
        strcat(buffer,p.branch);
        int x=strlen(buffer);
        for(int j=0;j<45-x;j++)
        strcat(buffer,"!");
        fp1<<count<<"|"<<buffer<<endl;
        fp2<<p.usn<<"|"<<count<<"|"<<endl;
        fp1.close();
        fp2.close();
}
void write()
{
        cout<<"Enter the name\n";
        cin>>t.name;
        cout<<"Enter the usn\n";
        cin>>t.usn;
        cout<<"Enter the age\n";
        cin>>t.age;
        cout<<"Enter the sem\n";
        cin>>t.sem;
        cout<<"Enter the branch\n";
        cin>>t.branch;
        pack(t);
}
void search()
{
        student x;
        fp1.open("std.txt",ios::in);
        fp2.open("ind.txt",ios::in);
```

```
        cout<<" Enter the USN to search for:";
        cin>>keyusn;
        for(i=0;i<count;i++)
        {

                fp2.getline(buffer,100);
                sscanf(buffer,"%[^|]|%[^|]|",tempusn ,temprrn);
                if(strcmp(tempusn,keyusn)==0)
                {
                        cout<<" Record found";
                        int rrn=atoi(temprrn);
                        cout<<" RRN value of record:"<<rrn;
                        fp1.seekg((rrn-1)*49,ios::beg);
                        fp1.getline(buffer1,100);
                        sscanf(buffer1,"%[^|]|%[^|]|%[^|]|%[^|]|%[^|]|%[^!]!",tempr
                        rn, x.name,x.usn,x.age,x.sem,x.branch);
                        cout<<"\nUSN:"<<x.usn<<"\nNAME:"<<x.name<<"\nage:
                        "<< x.age<<"\nsem:"<<x.sem<<"\nbranch:"<<x.branch;
                        break;
                }
        }
        if(i==count)
        cout<<"record not found";
        fp1.close();
        fp2.close();
}
void delet()
{
        int temp;
        if(i==count)
        return;
        fp1.open("std.txt",ios::in);
        for(int j=0;j<count;j++)
```

```
        {
                fp1.getline(buffer,100);
                sscanf(buffer,"%[^|]|%[^|]|%[^|]|%[^|]|%[^|]|%[^!]!",temp,s[j].name,
                s[j].usn,s[j].age,s[j].sem,s[j].branch);
        }
        fp1.close();
        remove("std.txt");
        remove("ind.txt");
        fp1.open("std.txt",ios::out);
        fp2.open("ind.txt",ios::out);
        fp1.close();
        fp2.close();
        int cntold=count;
        count=0;
        for(j=0;j<cntold;j++)
        if(j!=i)
        {
        count++;
        pack(s[j]);
        }
}
void main()
{
        int c;
        clrscr();
        fp1.open("std.txt",ios::out);
        fp2.open("ind.txt",ios::out);
        fp1.close();
        fp2.close();
        while(1)
        {
                cout<<"\n1.Write\n 2.Search\n 3.Delete\n 4.Exit\n Enter your choice\n";
```

```
            cin>>c;
            switch(c)
            {
                  case 1:count++;write();break;
                  case 2:search();break;
                  case 3:search();delet();break;
                  default:exit(0);
            }
      }
}
```

Output:

```
1. Write        2 .search       3. Delete       4. Exit
Enter your choice: 1
Enter the name : Sandesh
Enter the usn: 4su16IS020
Enter the age: 22
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .search       3. Delete       4. Exit
Enter your choice: 1
Enter the name : kiran
Enter the usn: 4su16is013
Enter the age: 21
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .search       3. Delete       4. Exit
Enter your choice: 1
Enter the name : Namratha
Enter the usn: 4su16is022
Enter the age: 20
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .search       3. Delete       4. Exit
Enter your choice: 2
Enter the usn : 4su16is022
```

```
Record found
RRN value of record :3
Name      usn          age   sem   branch
Namratha  4su16is022   20    6     ISE

1. Write      2 .search     3. Delete     4. Exit
Enter your choice: 3
Enter the usn : 4su16is013
Record found
RRN value of record :2
Name      usn          age   sem   branch
kiran     4su16is013   21    6     ISE

1. Write      2 .search     3. Delete     4. Exit
Enter your choice: 2
Enter the usn : 4su16is013
Record  not found

1. Write      2 .search     3. Delete     4. Exit
Enter your choice: 4
```

Viva Voce:

1. Define simple index.

   A simple is an index in which the entries are a key ordered linear list
2. What is multilevel indexing?

   Multilevel Indexing is created when a primary index does not fit in memory.
3. Explain the operations to maintain simple index file.
4. How simple index is different from RRN?
5. What is difference between scanf( ) and sscanf( )?

   Scanf reads from the standard input stream stdin.

   Sscanf reads from the character string.
6. Write syntax to read a list of record from a file.

   Open file.fptr = fopen (filename, "r")
7. Explain pack( ) and unpack( ) with respect to simple index?

   Simple index is used to create or read from the file.
8. Explain the different ways to write data into a file.

   Open file.fptr = fopen (filename, "w")

9. What are stdin, stdout and stderr?

   Standard Streams: Standard in, Standard out and Standard error respectively.

10. Explain how index file is created using primary key.

    Primary Indexing: These primary keys are unique to each record and contain 1:1 relation between the records

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

Algorithm:

Step 1: Start
Step 2: Open file std.txt and ind.txt
Step 3: create a file second.txt
Step 3: Enter the choice
1. Add 2. Search 3. Delete 4. Exit
case 1: add() //Read the name, usn, age, sem, branch
case 2: search()
case 3: search()
delet()
Step 3: Stop

Algorithm: add()
Step 1: Start
Step 2: Enter the name, usn, age, sem, branch
Step 3: update the second.txt with std.txt and ind.txt
Step 4: Stop

Algorithm: search()
Step 1: Start
Step 2: Enter the name of the student to be searched
Step 3: Open the second.txt and read the contents to buffer
Step 3: if(strcmp(key_name,sec_name)==0)
print Name found";
else
print as Record not found
Step 4: close file pointe fp1 and fp2
Step 5: Stop

Algorithm: delet()
Step 1: Start
Step 2: open a file std.txt
Step 3: scan the file and store the name, usn, age, sem, branch into buffer
Step 4: close a file
Step 5: remove the record from std.txt, ind.txt and second.txt
Step 6: Print record delete
Step 7: Stop

Program:

```
// Program to implement index on secondary key, the name, for a file of student objects
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
#include<fstream.h>
#include<stdlib.h>
#include<string.h>
class student
{
        public: char name[15];char usn[15]; char sem[5]; char age[5]; char branch[5];
};
student s[10],t;
char buf[50];
fstream fp1,fp2,fp3;
int count=0,x,i,j,l;
void add();
void search();
void delet();
void main()
{
        int ch;
        clrscr();
        fp1.open("std.txt",ios::out);
        fp2.open("ind.txt",ios::out);

        fp3.open("second.txt",ios::out);
        fp1.close();
        fp2.close();
        fp3.close();
        while(1)
        {
```

```
                cout<<"\n MENU:\n 1)add\n 2)search\n 3)delete\n 4)exit\n"<<endl;
                cout<<"enter your choice:";
                cin>>ch;
                switch(ch)
                {
                        case 1: count++; add();
                                break;
                        case 2: search();
                                break;
                        case 3: delet();
                                break;
                        case 4: exit(0);
                }
        }
}
void add()
{
        cout<<"enter name:";
        cin>>t.name;
        cout<<"enter usn:";
        cin>>t.usn;
        cout<<"enter sem:";
        cin>>t.sem;
        cout<<"enter age:";
        cin>>t.age;
        cout<<"enter branch:";
        cin>>t.branch;
        strcpy(buf,t.name);
        strcat(buf,"|");
        strcat(buf,t.usn);
        strcat(buf,"|");
        strcat(buf,t.sem);
```

```
        strcat(buf,"|");
        strcat(buf,t.age);
        strcat(buf,"|");
        strcat(buf,t.branch);
        x=strlen(buf);
        for(i=0;i<45-x;i++)
        strcat(buf,"!");
        fp1.open("std.txt",ios::app);
        fp2.open("ind.txt",ios::app);
        fp3.open("second.txt",ios::app);
        fp1<<count<<"|"<<buf<<endl;
        fp2<<t.usn<<"|"<<count<<"|"<<endl;
        fp3<<t.name<<"|"<<t.usn<<"|"<<endl;
        fp1.close();
        fp2.close();
        fp3.close();
}
void search()
{
        char key_usn[15], tempusn[15], temprrn[10], rrn,buf2[100], key_name[30],
        sec_usn[20], sec_name[20];
        cout<<"Enter the name of the student to be searched:";
        cin>>key_name;
        fp3.open("second.txt",ios::in);
        for(l=0;l<count;l++)
        {
                fp3.getline(buf2,100);
                sscanf(buf2,"%[^|]|%[^|]|",sec_name,sec_usn);
                if(strcmp(key_name,sec_name)==0)
                {
                        strcpy(key_usn,sec_usn);
                        cout<<"name is found:";
```

```
                break;
        }
}
fp3.close();
if(l==count)
{
        cout<<"name not found";
        i=count;
        return;
}
fp2.open("ind.txt",ios::in);
for(i=0;i<count;i++)
{
        fp2.getline(buf,100);
        sscanf(buf,"%[^|]|%[^|]|",tempusn, temprrn);
        if(strcmp(key_usn,tempusn)==0)
        {
                cout<<"record found"<<endl;
                cout<<"RRN of the record:"<<rrn<<endl;
                fp1.open("std.txt",ios::in);
                int rrnno=atoi(temprrn);
                fp1.seekg((rrnno-1)*49,ios::beg);
                fp1.getline(buf,100);
                sscanf(buf,"%[^|]|%[^|]|%[^|]|%[^|]|%[^|]|%[^!]!",rrn,t.name,t.usn,t.
                sem,t.age,t.branch);
                cout<<"name:"<<t.name<<endl;
                cout<<"usn:"<<t.usn<<endl;
                cout<<"sem:"<<t.sem<<endl;
                cout<<"age:"<<t.age<<endl;
                cout<<"branch:"<<t.branch<<endl;
                fp1.close();
                break;
```

```
                }
}
void delet()
{
        char temp[10];
        if(i==count)
        return;
        fp1.open("std.txt",ios::in);
        for(j=0;j<count;j++)
        {
                fp1.getline(buf,100);
                sscanf(buf,"%[^|]|%[^|]|%[^|]|%[^|]%[^|]|%[^!]!",temp,s[j].name,s[j].usn,s[
                j].sem,s[j].age,s[j].branch);
        }
        fp1.close();
        remove("std.txt");
        remove("ind.txt");
        remove("second.txt");
        fp1.open("std.txt", ios::out);
        fp2.open("ind.txt", ios::out);
        fp3.open("second.txt",ios::out);
        int oldcount=count;
        count=0;
        for(j=0;j<oldcount;j++)
        {
                if(j!=i)
                {
                        count++;
                        strcpy(buf,s[j].name);
                        strcat(buf,"|");
                        strcat(buf,s[j].usn);
                        strcat(buf,"|");
```

```
                    strcat(buf,s[j].sem);
                    strcat(buf,"|");
                    strcat(buf,s[j].age);
                    strcat(buf,"|");
                    strcat(buf,s[j].branch);
                    x=strlen(buf);
                    for(int k=0;k<45-x;k++)
                    strcat(buf,"!");
                    fp1<<count<<"|"<<buf<<endl;
                    fp2<<s[j].usn<<"|"<<count<<"|"<<endl;
                    fp3<<s[j].name<<"|"<<s[j].usn<<"|"<<endl;
                }
        }
        cout<<"record delete";
        fp1.close();
        fp2.close();
        fp3.close();
}
```

Output:

```
1. Write        2 .search       3. Delete       4. Exit
Enter your choice: 1
Enter the name : Sandesh
Enter the usn: 4su16is020
Enter the age: 22
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .search       3. Delete       4. Exit
Enter your choice: 1
Enter the name : kiran
Enter the usn: 4su16is013
Enter the age: 21
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .search       3. Delete       4. Exit
```

```
Enter your choice: 1
Enter the name : Namratha
Enter the usn: 4su16is022
Enter the age: 20
Enter the Sem: 6
Enter the branch : ISE

1. Write        2 .search       3. Delete       4. Exit
Enter your choice: 2
Enter the name of the student to be searched : Sandesh
Name is found
Record found
RRN of the record :1
Name    usn          age   sem   branch
Sandesh 4su16is020   22    6     ISE

1. Write        2 .search       3. Delete       4. Exit
Enter your choice: 3
Enter the usn : 4su16is013
Record found
RRN value of record :2
Name    usn          age   sem   branch
kiran   4su16is013   21    6     ISE

1. Write        2 .search       3. Delete       4. Exit
Enter your choice: 2
Enter the usn : 4su16is013
Record  not found

1. Write        2 .search       3. Delete       4. Exit
Enter your choice: 4
```

Viva Voce:

1. Define secondary index.

   The secondary Index is an indexing method whose search key specifies an order different from the sequential order of the file.

2. Distinguish between primary and secondary indexing

   A primary index is an index on a set of fields that includes the unique primary key and is guaranteed not to contain duplicates. In contrast, a secondary index is an index that is not a primary index and may have duplicates.

3. Explain how index file is created using secondary key.

4. How index file is different from primary and secondary key.

   The main difference between primary and secondary index is that the primary index is an index on a set of fields that includes the primary key and does not contain duplicates, while the secondary index is an index that is not a primary index and can contain duplicates.

5. What are the advantages and disadvantages of indexes that are too large to hold in memory?

   Advantages are:

   - It helps you to reduce the total number of I/O operations needed to retrieve that data.
   - Offers Faster search and retrieval of data to users.
   - Indexing also helps you to reduce tablespace.
   - You can't sort data in the lead nodes as the value of the primary key classifies it.

   Disadvantages are:

   - To perform the indexing database management system, you need a primary key on the table with a unique value.
   - You can't perform any other indexes on the Indexed data.
   - You are not allowed to partition an index-organized table.

6. What is Hashing? What is its use?

   Hashing is mapping a given value with a particular key for faster access of elements. One main use of hashing is to compare two files for equality.

7. What is the use of seekg() and seekp()?

   seekg() is used to move the get pointer to a desired location with respect to a reference point.

   seekp() is used to move the put pointer to a desired location with respect to a reference point.

8. What is File descriptor?

   A file descriptor is a number that uniquely identifies an open file in a computer's operating system.

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

Algorithm:

Step 1: Start

Step 2: Open the file f1.txt and f2.txt

Step 3: enter no. of names to read in file1 and file 2

Step 4: enter the names of file1 and file 2 in ascending order

Step 5: Open the file f3.txt

Step 6: Compare the elements of both the files f1, f2

if(strcmp(list1[i],list2[j])<0)

print match found, update the list on f3.txt

else

print not found

Step 7: Stop

Program:

```
// Program to read two lists of names and then match the names in the two lists using Consequential Match
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
#include<string.h>
void main()
{
    fstream fp1,fp2,fp3;
    char list1[50][20],list2[50][20];
    int n,m,i,j,k;
    fp1.open("f1.txt",ios::out);
    cout<<"enter no. of names to read in file1:"<<endl;
    cin>>n;
    cout<<"enter the names of file1 in ascending order:"<<endl;
```

```
for(i=0;i<n;i++)
{
        cin>>list1[i];
        fp1<<list1[i]<<endl;
}
fp1.close();
fp2.open("f2.txt",ios::out);
cout<<"enter no. of names to read in file2:"<<endl;
cin>>m;
cout<<"enter the names of file2 in ascending order:"<<endl;
for(i=0;i<m;i++)
{
        cin>>list2[i];
        fp2<<list2[i]<<endl;
}
fp2.close();
fp3.open("f3.txt",ios::out);
i=j=k=0;
while(i<n && j<m)
{
        if(strcmp(list1[i],list2[j])<0)
        i++;
        else if(strcmp(list1[i],list2[j])==0)
        {
                cout<<"names matched are:"<<list1[i]<<endl;
                fp3<<list1[i]<<endl;
                i++;
                j++;
                k++;
        }
        else
                j++;
```

```
        }
        cout<<"no. of match found:"<<k<<endl;
        fp3.close();
getch();
}
```

Output:

```
Enter the no. of names to read in file1
3
Enter the names of file1 in ascending order
Arpita
Darshan
Kiran
Enter the no. of names to read in file1
4
Enter the names of file1 in ascending order
Anusha
Divya
Kavya
Sandesh
No. of matches found : 0

Enter the no. of names to read in file1
3
Enter the names of file1 in ascending order
Arpita
Darshan
Kiran
Enter the no. of names to read in file1
4
Enter the names of file1 in ascending order
Anusha
Divya
Kiran
Sandesh
Names matched are : Kiran
No. of matches found : 1
```

Viva Voce:

1. Define consequential processing.

   Coordinated processing of two or more sequential lists to produce a single output list.

2. Where is the need of consequential technique?

   To reduce file access time.

3. What is Fragmentation?

   Fragmentation refers to the condition of a disk in which files are divided into pieces scattered around the disk.

4. What is data compression?

   Data compression is the process of modifying, encoding or converting the bits structure of data in such a way that it consumes less space on disk.

5. How can we reclaim the deleted space dynamically?

   Use an “avail list”, a linked list of available records.

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

Algorithm:

Step 1: Start

Step 2: Read the no of records

Step 3: Read the details (Considering 8 records)

Step 4: call kwaymerge() function

Step 5: stop

Algorithm: kwaymerge()

Step 1: Start

Step 2: call the merge_file function

```
merge_file("1.txt","2.txt","11.txt");
merge_file("3.txt","4.txt","22.txt");
merge_file("5.txt","6.txt","33.txt");
merge_file("7.txt","8.txt","44.txt");
merge_file("11.txt","22.txt","111.txt");
merge_file("33.txt","44.txt","222.txt");
merge_file("111.txt","222.txt","1111.txt");
```

Step 3: Stop

Algorithm: merge_file(file1, file2, file3)

Step 1: Start

Step 2: open file1, file2, file3

Step 3: read the names from file1 and file 2

Step 4: Compare the consecutive name from the files and merge the contents in third file

Step 5: Print the contents of all files

Step 6: Stop

Program:

```
// Program to read k lists of names and merge them using k-way merge
#include<iostream.h>
#include<fstream.h>
#include<string.h>
#include<conio.h>
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 *file3)
{
        char names[30][20];
        int k=0;
        fstream f1,f2,f3;
        f1.open(file1,ios::in);
        f2.open(file2,ios::in);
        f3.open(file3,ios::out);
        while(!f1.eof())
        {
                f1.getline(names[k],20,'\n');
                k++;
        }
        while(!f2.eof())
        {
                f2.getline(names[k],20,'\n');
                k++;
        }
        k=k-1;
        char temp[20];
        for(int x=0;x<k;x++)
        {
```

```
        for(int y=x+1;y<k;y++)
        {
                if(strcmp(names[x],names[y])>0)
                {
                        strcpy(temp,names[x]);
                        strcpy(names[x],names[y]);
                        strcpy(names[y],temp);
        }
        }
        }
        for(int i=1;i<k;i++)
        {
                f3<<names[i]<<endl;
        }
        f1.close();
        f2.close();
        f3.close();
}
void kwaymerge()
{
        merge_file("1.txt","2.txt","11.txt");
        merge_file("3.txt","4.txt","22.txt");
        merge_file("5.txt","6.txt","33.txt");
        merge_file("7.txt","8.txt","44.txt");
        merge_file("11.txt","22.txt","111.txt");
        merge_file("33.txt","44.txt","222.txt");
        merge_file("111.txt","222.txt","1111.txt");
        return;
}
void main()
{
        int i,no;
```

```
        char name[20];
        clrscr();
        cout<<"enter the no of records:";
        cin>>no;
        cout<<"enter the details:";

        for(i=0;i<8;i++)
        {
                file[i].open(fname[i],ios::out);
        }
        for(i=0;i<no;i++)
        {
                cout<<"NAME:";
                cin>>name;
                file[i%8]<<name<<endl;
        }
        file[i].close();
        kwaymerge();
        getch();
}
```

Output:

Enter no. of record: 14

Enter the details

Name: Anusha

Name:Arpita

Name:Darshan

Name:Divya

Name:Kavya

Name:Kiran

Name:Chethan

Name:Ganga

Name:Pavitra

Name:Koushik

Name:Karthik

Name:Ankitha

Name:Sandesh

Name:Vivek

1.txt	2.txt	3.txt	4.txt	5.txt	6.txt	7.txt	8.txt
Anusha	Darshan	Kavya	Chethan	Pavitra	Karthik	Sandesh	Vivek
Arpita	Divya	Kiran	Ganga	Koushik	Ankitha		

Viva Voce:

1. What is K-way merge algorithm?

   A merge sort that sorts a data stream using repeated merges. It distributes the input into k streams by repeatedly reading a block of input that fits in memory, called a run, sorting it, then writing it to the next stream. It merges runs from the k streams into an output stream.

2. What are the advantages of k-way merge algorithm?
   - It is very efficient and reliable.
   - It requires least amount of intervention.
3. What are the properties of B tree?
   - Every node has at most m children.
   - Every non-leaf node (except root) has at least ⌈m/2⌉ child nodes.
   - The root has at least two children if it is not a leaf node.
   - A non-leaf node with k children contains k − 1 keys.
4. What is an AVL tree?

   AVL tree is a self-balancing Binary Search Tree (BST) where the difference between heights of left and right subtrees cannot be more than one for all nodes.
5. What is the use of tellg() and tellp()?

   tellg() is used to know where the get pointer is in a file.

   tellp() is used to know where the put pointer is in a file.
6. What is Boeing tree?

   Boeing tree or B-tree is a self-balancing tree data structure that maintains sorted data and allows searches, sequential access, insertions, and deletions in logarithmic time.

Additional Programs

9. Write a C++ program to implement B+ tree for a given set of integers and its operations insert ( ), and search ( ). Display the tree.

Algorithm: Insertion

Step 1: Start

Step 2: Insert the new node as a leaf node

Step 3: If the leaf doesn't have required space, split the node and copy the middle node to the next index node.

Step 4: If the index node doesn't have required space, split the node and copy the middle element to the next index page.

Step 5: Stop

Algorithm: Deletion

Step 1: Start

Step 2: Delete the key and data from the leaves.

Step 3: if the leaf node contains less than minimum number of elements, merge down the node with its sibling and delete the key in between them.

Step 4: if the index node contains less than minimum number of elements, merge the node with the sibling and move down the key in between them.

Step 5: Stop

Algorithm: Search

Step 1: Start

Step 2: Searching a node in a B+ Tree

Step 3: Perform a binary search on the records in the current node.

Step 4: If a record with the search key is found, then return that record.

Step 5: If the current node is a leaf node and the key is not found, then search unsuccessful else repeat the process

Step 6: Stop

Program:

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

```
#include <conio.h>
#include <math.h>
#define M 4
#define MBY2 (int)ceil((float)M/2)
fstream file;
struct node {
        int level;
        int usage;
        int numbers[M+1];
        struct node* children[M+1];
        struct node* parent;
        struct node* right;
};
typedef struct node* NODE;
NODE getnode()
{
        NODE sumne;
        int i;
        sumne=(NODE)malloc(sizeof(struct node));
        if (sumne==NULL)
        {       cout<< "insuffcient memory\n";
                getch(); exit(0);
        }
        sumne->level=0; sumne->usage=0;
        for(i=0;i<=M;i++)
        {
                sumne->children[i]=NULL;
          sumne->numbers[i]=999;
        }
        sumne->parent=NULL;
        sumne->right=NULL;
        return sumne;
```

```
}
void display(NODE head)
{   NODE a[100],cur;
        int i=0,j=0,k,level;
        cur=head;
        a[i]=cur;i++;
        while (a[j]->children[0] != NULL)
        {       for(k=0;k<= (a[j]->usage)-1;k++)
                {       a[i]=a[j]->children[k] ;
                        i++;
                }
                j++;
        }
        i--;
        level=head->level;
        for(k=0;k<=i;k++)
        {       if (level != a[k]->level)
                { cout << "\n"; level = a[k]->level;}
                for(j=0;j<a[k]->usage;j++)
                if (j == a[k]->usage-1) cout << a[k]->numbers[j];
                else cout << a[k]->numbers[j] << "," ;
                cout << "\t" ;
        }
        cout << "\n";
}
void main()
{
        NODE head=NULL,cur,temp,up,prev,newup,first=NULL,sumne,sumsumne;
         char ch;
         int num,i,j,k,poi;
         clrscr();
         for(;;)
```

```
{   cout << "1:insert 2:search 3:displaytree 4:list_bottom 5:exit\n enter choice:";
    cin >> ch;
    switch(ch)
    {     case '1': cout << "enter number to insert into btree\n";
          cin >> num;
          if (head == NULL)
          {  head=getnode();
            head->usage=1;
            head->numbers[0]=num;
            first=head;
            break;
          }
          else
          {  cur=head;
            while (cur != NULL)
            {  for(i=0,j=0;i<=(cur->usage)-1;i++)
                    if (num > cur->numbers[i]) j++ ;
               prev=cur;
               if (j == cur->usage) cur=cur->children[j-1];
               else cur=cur->children[j];
            }
            cur=prev;
            poi=j;
            if (cur->numbers[poi] == num)
            {     cout << "element already present\n";
                  break;
            }
            if (poi != cur->usage)
            {  for(i=cur->usage;i>=poi+1;i--)
                  {  cur->numbers[i]=cur->numbers[i-1];
                    cur->children[i]=cur->children[i-1];
                  }
```

```
}
cur->numbers[poi]=num;
cur->children[poi]=NULL;
cur->usage++;
while(cur!=NULL)
{  if (cur->usage <= M)
      {   if (cur->parent == NULL) break;
          up=cur->parent; i=0;
          while (up->children[i] != cur) i++;
          up->numbers[i]=cur->numbers[(cur->usage)-1];
          cur=up;
          continue;
      }
      temp=getnode();
      for(k=MBY2,j=0;k<=M;k++,j++)
      {   temp->numbers[j]=cur->numbers[k];
          cur->numbers[k]=999;
          temp->children[j]=cur->children[k];
          cur->children[k]=NULL;
      }
      cur->usage=MBY2;
      temp->usage=M-(cur->usage)+1;
      temp->level=cur->level;
      if (cur->children[0] != NULL)
      {  for (i=0;i<=(cur->usage)-1;i++) (cur->children[i])->parent=cur;
        for (i=0;i<=(temp->usage)-1;i++)
              (temp->children[i])->parent=temp;
      }
      if (cur->level == 0)
      {  sumne=first;
        while (sumne->right != NULL && sumne != cur)
              sumne=sumne->right ;
```

```
    sumsumne=sumne->right;
    sumne->right=temp;
    temp->right=sumsumne;
  }
  if (cur->parent == NULL)
  {  up=getnode();
    up->level=(cur->level)+1;
    up->usage=2;
    up->numbers[0]=cur->numbers[(cur->usage)-1];
    up->numbers[1]=temp->numbers[(temp->usage)-1];
    up->children[0]=cur;
    up->children[1]=temp;
    cur->parent = temp->parent = up;
    head=up;
    break;
  }
  else
  {  up=cur->parent;
    temp->parent=up;
    newup=getnode();
    i=0;
    while (up->numbers[i] < cur->numbers[0])
    {   newup->numbers[i] = up->numbers[i];
          newup->children[i] = up->children[i];
          i++;
    }
    newup->numbers[i]= cur->numbers[(cur->usage)-1];
    newup->children[i] = cur;
    i++;
    newup->numbers[i]= temp->numbers[(temp->usage)-1];
    newup->children[i] = temp;
    i++;
```

```
                    j=0;
                    while (up->numbers[j] <= temp->numbers[(temp->usage)-1]) j++;
                    while (j <= (up->usage)-1)
                    {   newup->numbers[i] = up->numbers[j];
                        newup->children[i] = up->children[j];
                        i++,j++;
                    }
                    up->usage=i;
                    newup->usage=i;
                    for(i=0;i<=(newup->usage)-1;i++)
                    {     up->numbers[i] = newup->numbers[i];
                        up->children[i] = newup->children[i];
                    }
                    free(newup);
                    cur=up;
                    continue;
                  }
              }
              break;
case '2': cout << "enter number to search\n";
              cin >> num;
              cur=head;
              while (cur != NULL)
              {  for(i=0,j=0;i<=(cur->usage)-1;i++)
                     if (num > cur->numbers[i]) j++ ;
                prev=cur;
                if (j == cur->usage) cur=cur->children[j-1];
                else cur=cur->children[j];
              }
              cur=prev;
              poi=j;
```

```
                if (cur->numbers[poi] == num) cout << "search success\n";
                else cout << "search failure\n";
                break;
        case '3': display(head);
                break;
        case '4': sumne=first;
                while (sumne!=NULL)
                {       for(i=0;i<=(sumne->usage)-1;i++)
                        if ((sumne->right == NULL) && i == (sumne->usage)-1)
                        cout << sumne->numbers[i] ;
                        else cout << sumne->numbers[i] << ",";
                        sumne=sumne->right;
                }
                cout << "\n";
                break;
        default: exit(0);
     }
   }
}
```

Output:

```
Main menu
-----------------
    1. insert       2.search        3. Display tree     4.Exit
    enter your choice: 2
    enter the key:37
    the path traversed is:0-->0-->1-->
    element not found
Main menu
-----------------
    1. insert       2.search        3. Display tree     4.Exit
    enter your choice: 1
    enter the element:25
                    10          20          25
Main menu
-----------------
    1. insert       2.search        3. Display tree     4.Exit
    enter your choice: 1
```

```
    enter the element:85
                        80          85          90

Main menu
-----------------
    1. insert       2.search        3. Display tree     4.Exit
    enter your choice: 3
    level----------0
    50
    Level---------1
    30          70  80
    Level--------2
    10  20 25    30 40       50  60      70  75      80  85  90

Main menu
-----------------
    1. insert       2.search        3. Display tree     4.Exit
    enter your choice: 1
    enter the element:75
                        70      75      80      90
                        70      80      90
Main menu
-----------------
    1. insert       2.search        3. Display tree     4.Exit
    enter your choice: 1
    enter the element:90
                        50      70      90
```

10. Write a C++ program to store and retrieve student data from file using hashing. Use any collision resolution technique

Algorithm: Insert (string)

Step 1: Start

Step 2: Compute the index using Hash Function using index = hashFunc(s)

Step 3: Insert the element in the linked list at the particular index

Step 4: Stop

Algorithm: Search

Step 1: Start

Step 2: Compute the index by using the hash function

Step 3: search the linked list at that specific index

if(hashTable[index][i] == s)

print element found

else

print element not found

Step 4: Stop

Program:

// Write a C++ program to store and retrieve student data from file Using Hashing.

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

class node
{
        public:  char name[15];
                 char usn[15];
                 node * link;
};

node * h[29];

void insert()
{
        char name[15];
```

```
        char usn[15];
        char buffer[50];
        fstream out;
        out.open("student.txt",ios::app);
        if(!out)
        {
                cout<<"unable to open the file in append mode";
                getch();
                return;
        }
        cout<<"enter the name\n";
        cin>>name;
        cout<<"\nenter the usn\n";
        cin>>usn;

        strcpy(buffer,name);
        strcat(buffer,"|");
        strcat(buffer,usn);
        strcat(buffer,"\n");
        out<<buffer;
        out.close();
}

void hash_insert(char name1[],char usn1[],int hash_key)
{
        node * p;
        node * prev;
        node * curr;
        p=new node;

        strcpy(p->name,name1);
        strcpy(p->usn,usn1);
        p->link=NULL;
        prev=NULL;
        curr=h[hash_key];
        if(curr==NULL)
        {
                h[hash_key]=p;
                return;
        }
        while(curr!=NULL)
        {
                prev=curr;
                curr=curr->link;
        }
        prev->link=p;
}

void retrieve()
{
        fstream in;
        char name[15];
        char usn[15];
        int j;
        int count;
```

```
node * curr;
in.open("student.txt",ios::in);
if(!in)
{
        cout<<"unable to open the file in read mode\n";
        getch();
        exit(0);
}
while(!in.eof())
{
        in.getline(name,15,'|');
        in.getline(usn,15,'\n');
        count=0;
        for(j=0;j<strlen(usn);j++)
        {
                count=count+usn[j];
        }
        count=count%29;
        hash_insert(name,usn,count);
}
cout<<"\nenter the usn\n";
cin>>usn;
count=0;
for(j=0;j<strlen(usn);j++)
{
        count=count+usn[j];
}
count=count%29;
curr=h[count];
if(curr==NULL)
{
        cout<<"record not found";
        getch();
        return;
}
do
{
        if(strcmp(curr->usn,usn)==0)
        {
                cout<<"\nrecord found: "<<curr->usn<<" " <<curr->name;
                getch();
                return;
        }
        else
        {
                curr=curr->link;
        }
}while(curr!=NULL);

if(curr==NULL)
{
        cout<<"record not found";
        getch();
        return;
}
```

```
}

void main()
{
        int choice;
        clrscr();
        fstream out;
        out.open("student.txt",ios::out);
        if(!out)
        {
                cout<<"unable to open the file in append mode\n";
                getch();
                exit(0);
        }
        for(;;)
        {
                cout<<"\n1 : insert";
                cout<<"\n2 : retrieve";
                cout<<"\n3 : exit";
                cout<<"\nenter your choice";
                cin>>choice;
                switch(choice)
                {
                        case 1:  insert();
                                 break;

                        case 2:  retrieve();
                                 break;

                        case 3:  exit(0);

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

Output:

```
1: insert       2: retrieve       3. Exit
Enter your choice :1
Enter the name
Ambika
Enter the usn
1
1: insert       2: retrieve       3. Exit
Enter your choice :1
Enter the name
chethan
Enter the usn
2
1: insert       2: retrieve       3. Exit
Enter your choice :1
Enter the name
divya
```

```
Enter the usn
3
1: insert       2: retrieve       3. Exit
Enter your choice :1
Enter the name
mahesh
Enter the usn
4
1: insert       2: retrieve       3. Exit
Enter your choice :2
Enter the usn
4
record found is 4 mahesh
1: insert       2: retrieve       3. Exit
Enter your choice :2
Enter the usn
7
Record not found
1: insert       2: retrieve       3. Exit
Enter your choice :3
```