

PROGRAM 1

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

```
#include<iostream.h>
#include<fstream.h>
#include<string.h>
void reverse(char *name)
{
    int i,j,size;
    char temp;
    size=strlen(name);

    for(i=0,j=size-1;i<j;i++,j--)
    {
        temp=name[i];
        name[i]=name[j];
        name[j]=temp;
    }
}
void main(int argc,char *argv[])
{
    char name[30];
    if(argc==1)
    {
        do
        {
            cin>>name;
            reverse(name);
            cout<<name;
```

```
        if(cin.eof())
            break;
    }while(1);
}
else if(argc>2)
{
    fstream ip,fp;
    fp.open(argv[1],ios::in);
    ip.open(argv[2],ios::out|ios::app);

    do
    {
        fp>>name;
        reverse(name);
        cout<<name<<endl;
        ip<<name<<endl;
        if(fp.eof())
            break;
    }while(1);
    fp.close();
    ip.close();
}
else
    cout<<"error";
}
```

Output1:

Vijay Kumar

yajiVramuK

BIT

TIB

Bangalore
erolagnaB

Output2:

D:\BIT ISE DEPT\LABS\fwdfslabprograms\Debug>1 abc.txt xyz.txt

avihS

ramuK

yajiV

TIB

Note : abc.txt should be edited by the student before execution. The output file xyz.txt must be verified after execution

PROGRAM 2:-

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

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

class FLB;

class person
{
    char usn[30],name[30],addr[30],branch[30],colg[30];
public:
    void input();
    void output();
    void search(char *fname);
    void modify(char *fname);
    friend class FLB;
};

class FLB
{
    char buff[160];
public:
    FLB()
    {
        for(int i=0;i<160;i++)
```

```
        buff[i]='\0';
    }
    void pack(person &p);
    void unpack(person &p);
    void read(fstream &fs);
    void write(char *fname);
};

void person::input()
{
    cout<<"enter usn,name,address,branch and college:";
    cin>>usn>>name>>addr>>branch>>colg;
}

void person::output()
{
    cout<<"usn: "<<usn<<"\nname: "<<name<<"\naddress: "<<addr<<"\nbranch:
"<<branch<<"\ncollege: "<<colg;
}

void FLB::pack(person &p)
{
    strcpy(buff,p.usn);strcat(buff,"|");
    strcat(buff,p.name);strcat(buff,"|");
    strcat(buff,p.addr);strcat(buff,"|");
    strcat(buff,p.branch);strcat(buff,"|");
    strcat(buff,p.colg);
}

void FLB::unpack(person &p)
```

```
{  
    char *ptr=buff;  
    while(*ptr)  
    {  
        if(*ptr=='|')  
            *ptr='\0';  
        ptr++;  
    }  
    ptr=buff;  
    strcpy(p.usn,ptr);  
    ptr+=strlen(ptr)+1;  
    strcpy(p.name,ptr);  
    ptr+=strlen(ptr)+1;  
    strcpy(p.addr,ptr);  
    ptr+=strlen(ptr)+1;  
    strcpy(p.branch,ptr);  
    ptr+=strlen(ptr)+1;  
    strcpy(p.colg,ptr);  
}  
void FLB::read(fstream &fs)  
{  
    fs.read(buff,sizeof(buff));  
}  
void FLB::write(char*fname)  
{  
    fstream os(fname,ios::app);
```

```
        os.write(buff,sizeof(buff));

        os.close();
    }

void person::search(char * fname)
{
    int found=0;

    person p;

    FLB b;

    char key[30];

    fstream is(fname,ios::in);

    cout<<"enter the usn to be searched:";

    cin>>key;

    while((!is.eof())&&(!found))
    {
        b.read(is);

        if(is.eof())
            break;

        b.unpack(p);

        if(strcmp(p.usn,key)==0)
        {
            cout<<"record found!!\n";


            p.output();

            found=1;
        }
    }

    if(!found)
```

```
        cout<<"record with given usn does't exist\n";
    is.close();
}

void person::modify(char*fname)
{
    int found=0;
    person p;
    FLB b;
    char key[30],tname[]="temp.txt";
    fstream is(fname,ios::in);
    fstream tfile(tname,ios::out|ios::app);
    cout<<"enter the usn of record to be modified:";
    cin>>key;
    while(!is.eof())
    {
        b.read(is);
        if(is.eof())
            break;
        b.unpack(p);
        if(strcmp(p.usn,key)==0 && !found)
        {
            cout<<"record found!!\n";
            p.input();
            found=1;
        }
        b.pack(p);
```




```
b.write(tname);
}

if(!found)
    cout<<"record with given usn does't exist\n";
is.close();
tfile.close();
remove(fname);
rename(tname,fname);
}

void main()
{
    int ch;
    person p;
    FLB b;
    char fname[]="prg2.txt";
    do
    {
        cout<<"1:insert\t2:search\t3:modify\t4:exit\nenter your choice";
        cin>>ch;
        switch(ch)
        {
            case 1:p.input();
                b.pack(p);
                b.write(fname);
                break;
```

should be written whenever we give input or modify

```
        case 2:p.search(fname);
            break;
        case 3:
            p.modify(fname);
            break;
        case 4://remove(fname);
            break;
    }
}while(ch!=4);
}
```

Output:

1:insert 2:search 3:modify 4:exit

enter your choice1

enter usn,name,address,branch and college:101 Shivakumar Bangalore ISE BIT

1:insert 2:search 3:modify 4:exit

enter your choice 1

enter usn,name,address,branch and college: 202 Vijay Hassan CSE MCE

1:insert 2:search 3:modify 4:exit

enter your choice 2

enter the usn to be searched:202

record found!!

usn: 202

name: Vijay

address: Hassan

branch: CSE

college: MCE

1:insert 2:search 3:modify 4:exit

enter your choice 3

enter the usn of record to be modified:101

record found!!

enter usn,name,address,branch and college:300 Arnold Newyork CSE BIT

1:insert 2:search 3:modify 4:exit

enter your choice 2

enter the usn to be searched:300

record found!!

usn: 300

name: Arnold

address: Newyork

branch: CSE

college: BIT1:insert 2:search 3:modify 4:exit

PROGRAM 3 :-

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

```
#include<fstream.h>
#include<stdio.h>
#include<string.h>
int n, no_of_rec, length, len[20];

class student
{
public:
    char usn[20], sem[20], name[20], branch[20];
};

class textbuff
{
public:
    char buff[500];
    void pack(student s[])
    {
        fstream file1("first.txt", ios::out | ios::app);
        clear_buff();
        for(int i = 0; i < n; i++)
        {
            strcat(buff, s[i].usn);
```

```
        strcat(buff, "|");
        strcat(buff, s[i].name);
        strcat(buff, "|");
        strcat(buff, s[i].branch);
        strcat(buff, "|");
        strcat(buff, s[i].sem);
        strcat(buff, "|");
        no_of_rec++;
        len[no_of_rec] = strlen(buff) + length;
    }
    length = strlen(buff);
    file1<<buff;
    file1.close();
}

void unpack(student s[])
{
    fstream file2("first.txt", ios::in);
    clear_buff();
    for(int i = 0; i < no_of_rec; i++)
    {
        file2.getline(s[i].usn, 20, '|');
        file2.getline(s[i].name, 20, '|');
        file2.getline(s[i].branch, 20, '|');
        file2.getline(s[i].sem, 20, '|');
    }
}
```

```
void clear_buff()
```

```
{
```

```
    for(int i = 0; i < 500; i++)
```

```
        buff[i] = NULL;
```

```
}
```

```
void display(student s[])
```

```
{
```

```
    char ch;
```

```
    if(no_of_rec != 0)
```

```
    {
```

```
        cout<<"USN\tName\tBranch\tSem\n";
```

```
        for(int i = 0; i < no_of_rec; i++)
```

```
        {
```

```
            ch = s[i].usn[0];
```

```
            if(ch != '*')
```

```
                cout<<s[i].usn<<"\t"<<s[i].name<<"\t"<<s[i].branch<<"\t"<<s[i].sem<<endl;
```

```
            }
```

```
        }
```

```
    else
```

```
        cout<<"No record found\n";
```

```
}
```

```
void search(student s[])
```

```
{
```

```
    int flag = 0;
```

```
    char key[20];
```

```
clear_buff();

cout<<"Enter the USN for search: ";

cin>>key;

for(int i = 0; i < no_of_rec; i++)
{
    if(strcmp(key, s[i].usn) == 0)
    {
        cout<<"Record found!!\n";

        cout<<"USN: "<<s[i].usn<<", Name: "<<s[i].name<<", Branch:
"<<s[i].branch<<", Sem: "<<s[i].sem<<endl;

        flag = 1;
    }
}

if(!flag)
    cout<<"No such USN found\n";
}

void modify(student s[])
{
    char key[20], cvar[20];

    /*int len1[20];*/

    int flag = 0, len_field;

    /*for(int i = 1; i <= no_of_rec; i++)

        len1[i] = len[i] - len[i-1];*/

    cout<<"Enter the USN of record to be modified: ";

    cin>>key;

    for(int i = 0; i < no_of_rec; i++)
```

```
{
    if(strcmp(key, s[i].usn) == 0)
    {
        fstream file3("first.txt", ios::in | ios::out);
        fstream file4("first.txt", ios::out | ios::app);
        file3.seekg(len[i], ios::beg);
        file3.getline(cvar, 20, '|');
        len_field = strlen(cvar); //length of 1 field
        file3.seekp(len[i], ios::beg);
        for(int j = 0; j < len_field; j++)
            file3<<'*';
        file3.close();
        cout<<"Enter new data:\nEnter USN, name, branch and sem:\n";
        cin>>s[i].usn>>s[i].name>>s[i].branch>>s[i].sem;
        clear_buff();
        strcat(buff, s[i].usn);
        strcat(buff, "|");
        strcat(buff, s[i].name);
        strcat(buff, "|");
        strcat(buff, s[i].branch);
        strcat(buff, "|");
        strcat(buff, s[i].sem);
        strcat(buff, "|");
        no_of_rec++;
        len[no_of_rec] = strlen(buff) + length;
        //      file3.seekp(0, ios::end);
    }
}
```



```
        file4<<buff;
        file4.close();
        flag = 1;
    }
}
if(!flag)
    cout<<"No such USN found\n";
}
};

void main()
{
    int ch, i;
    student s[20];
    textbuff b;
    cout<<"1.Create\n2.Display\n3.Search\n4.Modify\n5.Exit\n";
    do
    {
        cout<<"Enter your choice: ";
        cin>>ch;
        switch(ch)
        {
            case 1: cout<<"Enter no. of records: ";
                    cin>>n;
                    for(i = 0; i < n; i++)
                    {
                        cout<<"Enter USN, name, branch and sem of student "<<i+1<<endl;
```

```
        cin>>s[i].usn>>s[i].name>>s[i].branch>>s[i].sem;
    }

    b.pack(s);
    break;
case 2: b.unpack(s);
        b.display(s);
        break;
case 3: b.unpack(s);
        b.search(s);
        break;
case 4: b.unpack(s);
        b.modify(s);
        break;
case 5: remove("first.txt");
        break;
    }
}while(ch != 5);
}
```

Output:-

Enter USN, name, branch and sem of student 3

300 SauravG ECE 6

Enter USN, name, branch and sem of student 4

400 Kapil Mech 8

Enter your choice: 2

USN	Name	Branch	Sem
-----	------	--------	-----

100 Sachin ISE 6

200 R.Dravid CSE 6

300 SauravG ECE 6

400 Kapil Mech 8

Enter your choice: 3

Enter the USN for search: 400

Record found!!

USN: 400, Name: Kapil, Branch: Mech, Sem: 8

Enter your choice: 4

Enter the USN of record to be modified: 100

Enter new data:

Enter USN, name, branch and sem:

500 Gavaskar IT 9

Enter your choice: 2

USN	Name	Branch	Sem
-----	------	--------	-----

200	R.Dravid	CSE	6
-----	----------	-----	---

300	SauravG	ECE	6
-----	---------	-----	---

400	Kapil	Mech	8
-----	-------	------	---

500	Gavaskar	IT	9
-----	----------	----	---

PROGRAM 4 :-

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

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

class delimtextbuffer;

class person
{
    char usn[20];
    char name[18];
    char address[20];
    char branch[20];
    char college[20];
public:
    void input();
    void output();
    friend class delimtextbuffer;
};

class delimtextbuffer
{
    char buffer[160];
    char delim;
public:
    void pack(person& p);
    void unpack(person& p);
    void Read(fstream& fs);
```

```
        int Write(char *filename);
        delimtextbuffer();
};
class operations
{
    char *rrnfilename;
    char *recordfilename;
public:
    int maxrecords();
    operations(char *rrnfile,char *recordfile);
    void search();
    void insert();
};
operations::operations(char *rrnfile,char *recordfile)
{
    rrnfilename=rrnfile;
    recordfilename=recordfile;
}
int operations::maxrecords()
{
    fstream file(rrnfilename,ios::in);
    int pos;
    file.seekg(0,ios::end);
    pos=file.tellg();
    return (pos/(sizeof(int)));
}
void person::input()
{
    cout<<"Enter usn"<<endl;
    cin>>usn;
    cout<<"Enter name"<<endl;
    cin>>name;
```

```
        cout<<"Enter address"<<endl;
        cin>>address;
        cout<<"Enter branch"<<endl;
        cin>>branch;
        cout<<"Enter college"<<endl;
        cin>>college;
    }
    void person::output()
    {
        cout<<"USN:";
        puts(usn);
        cout<<"Name:";
        puts(name);
        cout<<"Address:";
        puts(address);
        cout<<"Branch:";
        puts(branch);
        cout<<"College:";
        puts(college);
    }
    delimtextbuffer::delimtextbuffer()
    {
        for(int i=0;i<160;i++)
            buffer[i]='\0';
        delim='|';
    }
    void delimtextbuffer::pack(person &p)
    {
        strcpy(buffer, p.usn);
        strcat(buffer, "|");
        strcat(buffer, p.name);
        strcat(buffer, "|");
```

```
        strcat(buffer, p.address);
        strcat(buffer, "|");
        strcat(buffer, p.branch);
        strcat(buffer, "|");
        strcat(buffer, p.college);
        strcat(buffer, "|");
        strcat(buffer, "*");
    }

void delimtextbuffer::unpack(person &p)
{
    char *ptr=buffer;
    while(*ptr)
    {
        if(*ptr == '|')
            *ptr = '\0';
        ptr++;
    }
    ptr = buffer;
    strcpy(p.usn, ptr);
    ptr += strlen(ptr) + 1;
    strcpy(p.name, ptr);
    ptr += strlen(ptr) + 1;
    strcpy(p.address, ptr);
    ptr += strlen(ptr) + 1;
    strcpy(p.branch, ptr);
    ptr += strlen(ptr) + 1;
    strcpy(p.college, ptr);
}

void delimtextbuffer::Read(fstream& fs)
{
    fs.getline(buffer,160,'*');
}
```

```
int delimtextbuffer::Write(char *filename)
{
    fstream os(filename,ios::out|ios::app);
    os.seekg(0,ios::end);
    int offset=os.tellp();
    os.write(buffer,strlen(buffer));
    os.close();
    return offset;
}

void operations::search()
{
    int rrn;
    cout<<"enter the RRN of the record to be searched(1 - based)";
    cin>>rrn;
    if(rrn>maxrecords()||rrn<=0)
    {
        cout<<"record not found!!!"<<endl;
        return;
    }
    delimtextbuffer b;
    person p;
    int offset=8;
    fstream file(rrnfilename,ios::in);

    file.seekg((rrn-1)*sizeof(int),ios::beg);
    file.read((char*)&offset,sizeof(offset));

    file.close();
    cout<<"record found!!!"<<endl;
    fstream file1(recordfilename,ios::in);
    file1.seekg(offset,ios::beg);
    b.Read(file1);
}
```



```
        b.unpack(p);
        p.output();
        file1.close();
    }
void operations::insert()
{
    person ob;
    delimittextbuffer b;
    int offset;
    fstream file,file2;
    ob.input();
    b.pack(ob);
    offset=b.Write(recordfilename);
    file.open(rrnfilename,ios::out|ios::app);
    file2.open("rr.txt",ios::out|ios::app);
    file2<<offset;
    file.write((char*)&offset,sizeof(int));
    file.close();
}
int main()
{
    int choice=1;
    fstream file,file1;
    person ob;
    delimittextbuffer b;
    char filename[20]="name.txt";
    char rrnfilename[20]="rrn.txt";
    operations o(rrnfilename,filename);
    while(choice<3)
    {
        cout<<"1:Insert a Record"<<endl;
        cout<<"2:Search for a Record"<<endl;
```

```
        cout<<"3:exit"<<endl;
        cin>>choice;
        switch(choice)
        {
        case 1:o.insert();
            break;
        case 2:o.search();
            break;
        }
    }
    return 0;
}
```

Output:

1:Insert a Record

2:Search for a Record

3:exit

1

Enter usn

100

Enter name

Darshan

Enter address

Vijayanagar

Enter branch

ISE

Enter college

BIT

1:Insert a Record

2:Search for a Record

3:exit

1

Enter usn

200

Enter name

Bhimsen

Enter address

Pune

Enter branch

CSE

Enter college

MIT

1:Insert a Record

2:Search for a Record

3:exit

2

enter the RRN of the record to be searched(1 - based)2

record found!!

200

Bhimsen

Pune

CSE

MIT

USN:Name:Address:Branch:College:1:Insert a Record

2:Search for a Record

3:exit

Program 5 :- Write a C++ program to implement simple index on primary key for a file of student objects. Implement the index.

```
#include<fstream.h>
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
char index[100][50],indexadd[100][50];
int nor;
void smi()
{
    char temp[100],temp1[100];
    for(int i=0;i<nor;i++)
    {
        for(int j=i+1;j<nor;j++)
        {
            if(strcmp(index[i],index[j])>0)
            {
                strcpy(temp,index[i]);
                strcmp(index[i],index[j]);
                strcmp(index[j],temp);
                strcmp(temp1,indexadd[i]);
                strcmp(indexadd[i],indexadd[j]);
                strcmp(indexadd[j],temp1);
            }
        }
    }
}
```

```
class stud
{
public:char usn[10],name[20],brn[10],sem[10];

    void unpack(int);
    void add();
    void del();
    void search();
    void disp();
};

void stud::add()
{
    char buf[100],sadd[50];
    cout<<"\n enter usn,name,branch,and sem:\n";
    cin>>usn>>name>>brn>>sem;
    for(int i=0;i<nor;i++)
    {
        if(strcmp(usn,index[i])==0)
        {
            cout<<"\nduplicate entry\n";
            return;
        }
    }
    strcpy(buf,usn);
    strcat(buf,"|");
    strcat(buf,name);
    strcat(buf,"|");
    strcat(buf,brn);
    strcat(buf,"|");
    strcat(buf,sem);
```

```
    strcat(buf,"|");
    fstream f1("index.txt",ios::app);
    fstream f2("rec.txt",ios::app);
    f2.seekp(0,ios::end);
    strcpy(index[nor],usn);

    itoa(f2.tellp(),sadd,10);
    strcpy(indexadd[nor],sadd);
    f1<<usn<<'|'<<f2.tellp()<<'\n';
    f2<<buf;
    nor++;
    smi();
    f1.close();
    f2.close();
}

void stud::search()
{
    char usn[10];
    stud s;
    cout<<"\nenter usn for search:";
    cin>>usn;
    int max=nor-1,min=0,flag=0,add,mid;
    while(min<=max&&flag==0)
    {
        mid=(min+max)/2;
        if(strcmp(usn,index[mid])==0)
        {
            flag=1;
            add=atoi(indexadd[mid]);
        }
    }
}
```

```
s.unpack(add);
cout<<"record found\n";
cout<<"usn:"<<s.usn<<"\tname:"<<s.name<<"\tbranch:"<<s.brn<<"\tsem:"<<s.sem<<endl;
    }
else
    if(strcmp(usn,index[mid])>0)
        min=mid+1;
    else
        max=mid-1;
}
if(!flag)
    cout<<"\nsearch failed!!\n";
}

void stud::unpack(int add)
{
    fstream f1("rec.txt",ios::in);
    f1.seekg(add);
    f1.getline(usn,10,'|');
    f1.getline(name,10,'|');
    f1.getline(brn,10,'|');
    f1.getline(sem,10,'|');
    f1.close();
}

void stud::del()
{
    char usn[10];
    fstream f1("index.txt",ios::out|ios::trunc);
    cout<<"\nenter usn to delete:";
    cin>>usn;
```

```
int max=nor-1,min=0,flag=0,mid;
while(min<=max&&flag==0)
{
    mid=(min+max)/2;
    if(strcmp(usn,index[mid])==0)
    {
        flag=1;
        cout<<"\nrecord delated\n";
        for(int i=mid;i<nor;i++)
        {
            strcpy(index[i],index[i+1]);
            strcpy(indexadd[i],indexadd[i+1]);
        }
        nor--;
    }
    else
        if(strcmp(usn,index[mid])>0)
            min=mid+1;
        else
            max=min-1;
}
for(int i=0;i<nor;i++)
{
    f1.write(index[i],strlen(index[i]));
    f1.write("|",1);
    f1.write(indexadd[i],strlen(indexadd[i]));
    f1.write("\n",2);
}
f1.close();
```



```
        if(!flag)
            cout<<"\ndeletion failed\n";
    }
void stud::disp()
{
    stud s;
    int add;
    cout<<"usn\tname\tbranch\tsem\n";
    for(int i=0;i<nor;i++)
    {
        add=atoi(indexadd[i]);
        s.unpack(add);
        cout<<s.usn<<"\t"<<s.name<<"\t"<<s.brn<<"\t"<<s.sem<<endl;
    }
}
void main()
{
    stud s;
    int ch;
    cout<<"1 add\n2 search\n3 delete\n4 display\n5 exit\n";
    do
    {
        cout<<"enter your chaice:";
        cin>>ch;
        switch(ch)
        {
            case 1:s.add();
                break;
            case 2:s.search();
```

```
        break;
    case 3:s.del();
        break;
    case 4:s.disp();
        break;
    case 5:remove("index.txt");
        remove("rec.txt");
        break;
    }
}while(ch!=5);
}
```

OUTPUT:-

1 add

2 search

3 delete

4 display

5 exit

enter your choice:1

enter usn,name,branch,and sem:

100 Ramya ISE 6

enter your choice:1

enter usn,name,branch,and sem:

200 Pooja CSE 6

enter your choice:2

enter usn for search:Pooja

search failed!!

enter your choice:2

enter usn for search:200

record found

usn:200 name:Pooja branch:CSE sem:6

enter your choice:3

enter usn to delete:200

record delated

enter your choice:4

usn name branch sem

100 Ramya ISE 6

enter your choice:

Program 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<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<fstream.h>

class record
{
public:
    char usn[10],name[20],branch[10],sem[2];
}rec[20],found[20];

int m=0,n;

void add()
{
    int i;
    cout<<"Enter number of students: ";
    cin>>n;
    cout<<"Enter Name, USN, Branch and Sem\n";
    n=m+n;
    for(i=m;i<n;i++)
    {
        cout<<"Student "<<i+1<<" : \n";
        cin>>rec[i].name>>rec[i].usn>>rec[i].branch>>rec[i].sem;
        m++;
    }
}
```

```
void sort_records()
{
    int i,j;
    record temp;
    for(i=0;i<n-1;i++)
    {
        for(j=0;j<n-i-1;j++)
        {
            if(strcmp(rec[j].name,rec[j+1].name)>0)
            {
                temp=rec[j];
                rec[j]=rec[j+1];
                rec[j+1]=temp;
            }
            else if(strcmp(rec[j].name,rec[j+1].name)==0)
            {
                if(strcmp(rec[j].usn,rec[j+1].usn)>0)
                {
                    temp=rec[j];
                    rec[j]=rec[j+1];
                    rec[j+1]=temp;
                }
            }
        }
    }
}
```

```
void create_file()
{
    ofstream index("secindex.txt",ios::out);
    ofstream file("record.txt",ios::out);
```

```
for(int i=0;i<n;i++)
{
    index<<rec[i].name<<"|"<<rec[i].usn<<"|"<<i<<"\n";

    file<<i<<"|"<<rec[i].usn<<"|"<<rec[i].name<<"|"<<rec[i].branch<<"|"<<rec[i].sem<<"\n";
}
index.close();
file.close();
}
```

```
void retrieve_record(char *index)
{
    fstream f1("record.txt",ios::in);
    int i;
    char ind[3],usn[10],name[20],branch[10],sem[2];
    for(i=0;i<n;i++)
    {
        f1.getline(ind,3,'|');
        f1.getline(usn,10,'|');
        f1.getline(name,20,'|');
        f1.getline(branch,10,'|');
        f1.getline(sem,2,'\n');
        if(strcmp(index,ind)==0)
            cout<<"USN: "<<usn<<" Name: "<<name<<" Branch: "
            <<branch<<" Sem: "<<sem<<endl;
        f1.seekg(2,ios::cur);
    }
    f1.close();
}
```

```
void search()
```

```
{
    int k=0,i,flag;
    fstream f1("secindex.txt",ios::in);
    char name[20],usn[10],ind[3],key_usn[10],key_name[20];
    char index[20][20];
    cout<<"Enter the name for search: ";
    cin>>key_name;
    for(i=0;i<n;i++)
    {
        f1.getline(name,20,'|');
        f1.getline(usn,10,'|');
        f1.getline(ind,3,'\n');
        if(strcmp(key_name,name)==0)
        {
            strcpy(found[k].name,name);
            strcpy(found[k].usn,usn);
            strcpy(index[k],ind);
            k++;
        }
    }
    f1.close();
    if(k==0)
        cout<<"Search failed!!\n";
    else if(k==1)
        retrieve_record(index[0]);
    else
    {
        cout<<"Choose USN from the list:\n";
        for(i=0;i<k;i++)
            cout<<found[i].usn<<endl;
        cin>>key_usn;
        flag=0;
```

```
        for(i=0;i<k;i++)
        {
            if(strcmp(key_usn,found[i].usn)==0)
            {
                retrieve_record(index[i]);
                flag=1;
            }
        }
        if(!flag)
            cout<<"Invalid entry!!\n";
    }
}
```

```
void delete_record(char *indx)
{
    int i,flag;
    char ch;
    fstream f1("record.txt",ios::in);
    char index[20][20];
    for(i=0;i<n;i++)
    {
        f1.getline(index[i],3,'|');
        f1.getline(rec[i].usn,10,'|');
        f1.getline(rec[i].name,20,'|');
        f1.getline(rec[i].branch,10,'|');
        f1.getline(rec[i].sem,2,'\n');
        f1.seekg(2,ios::cur);
        cout<<f1.tellg()<<endl;
    }
    flag=-1;
    for(i=0;i<n;i++)
        if(strcmp(index[i],indx)==0)
```



```
        flag=i;
    if(flag!=n-1)
    {
        for(i=flag;i<n;i++)
            rec[i]=rec[i+1];
    }
    n--;
    cout<<"Record deleted!!\n";
    f1.close();
    f1.open("secindex.txt",ios::out);
    fstream f2("record.txt",ios::out);
    for(i=0;i<n;i++)
    {
        f1<<rec[i].name<<"|"<<rec[i].usn<<"|"<<i<<"\n";

        f2<<i<<"|"<<rec[i].usn<<"|"<<rec[i].name<<"|"<<rec[i].branch<<"|"<<rec[i].sem
<<"\n";
    }
    f1.close();
    f2.close();
}

void del()
{
    int k=0,i,flag;
    fstream f1("secindex.txt",ios::in);
    char name[20],usn[10],ind[3],key_usn[10],key_name[20];
    char index[20][20];
    cout<<"Enter the name to delete: ";
    cin>>key_name;
    for(i=0;i<n;i++)
    {
        f1.getline(name,20,"|");
```

```
f1.getline(usn,10,'|');
f1.getline(ind,3,'\n');
if(strcmp(key_name,name)==0)
{
    strcpy(found[k].name,name);
    strcpy(found[k].usn,usn);
    strcpy(index[k],ind);
    k++;
}
}
f1.close();
if(k==0)
    cout<<"Deletion failed!!\n";
else if(k==1)
    delete_record(index[0]);
else
{
    cout<<"Choose USN from the list:\n";
    for(i=0;i<k;i++)
        cout<<found[i].usn<<endl;
    cin>>key_usn;
    flag=0;
    for(i=0;i<k;i++)
    {
        if(strcmp(key_usn,found[i].usn)==0)
        {
            delete_record(index[i]);
            flag=1;
        }
    }
    if(!flag)
        cout<<"Invalid entry!!\n";
}}
```

```
void main()
{
    int ch;
    cout<<"1:Add\n2:Search\n3:Delete\n4:Exit\n";
    do
    {
        cout<<"\nEnter your choice: ";
        cin>>ch;
        switch(ch)
        {
            case 1:
                add();
                sort_records();
                create_file();
                break;

            case 2:
                search();
                break;

            case 3:
                del();
                break;

            case 4:
                remove("secindex.txt");
                remove("record.txt");
                break;
        }
    }while(ch!=4);
}
```

OUTPUT:-

1:Add
2:Search
3>Delete
4:Exit

Enter your choice: 1

Enter number of students: 2

Enter Name, USN, Branch and Sem

Student 1 :

Abc 100 ise 2

Student 2 :

Xyz 200 ise 6

Enter your choice: 2

Enter the name for search: xyz

USN: 200 Name: xyz Branch: ise Sem: 6

Enter your choice: 3

Enter the name to delete: abc

Record deleted!!

Enter your choice: 2

Enter the name for search: abc

Search failed!!

Enter your choice: 2

Enter the name for search: xyz

USN: 200 Name: xyz Branch: ise Sem: 6

PROGRAM 7:-

Write a C++ 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.

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

void write_file()
{
    char name[20][20],temp[20];
    fstream f1("file1.txt",ios::out);
    fstream f2("file2.txt",ios::out);
    int i,j,m,n;
    if((!f1)||(!f2))
    {
        cout<<"Unable to open the files!!\n";
        exit(0);
    }
    cout<<"Enter the number of records in file 1: ";
    cin>>m;
    cout<<"Enter the names:\n";
    for(i=0;i<m;i++)
        cin>>name[i];
    for(i=0;i<m-1;i++)
    {
        for(j=0;j<m-i-1;j++)
        {
            if(strcmp(name[j],name[j+1])>0)
```

```
        {
            strcpy(temp,name[j]);
            strcpy(name[j],name[j+1]);
            strcpy(name[j+1],temp);
        }
    }
}

for(i=0;i<m;i++)
    f1<<name[i]<<endl;
cout<<"Enter the number of records in file 2: ";
cin>>n;
cout<<"Enter the names:\n";
for(i=0;i<n;i++)
    cin>>name[i];
for(i=0;i<n-1;i++)
{
    for(j=0;j<n-i-1;j++)
    {
        if(strcmp(name[j],name[j+1])>0)
        {
            strcpy(temp,name[j]);
            strcpy(name[j],name[j+1]);
            strcpy(name[j+1],temp);
        }
    }
}

for(i=0;i<n;i++)
    f2<<name[i]<<endl;
f1.close();
f2.close();
}
```

```
void main()
{
    fstream f1("file1.txt",ios::in);
    fstream f2("file2.txt",ios::in);
    fstream f3("output.txt",ios::out);
    char l1[100][20],l2[100][20],l3[100][20];
    int i=0,j=0,k=0,m=0,n=0;
    write_file();
    if((!f1)||(!f2)||(!f3))
    {
        cout<<"Unable to open the files!!\n";
        exit(0);
    }
    cout<<"Contents of file 1 are:\n";
    while(!f1.eof())
    {
        f1.getline(l1[i],20,'\n');
        cout<<l1[i++]<<endl;
        m++;
    }
    cout<<"Contents of file 2 are:\n";
    while(!f2.eof())
    {
        f2.getline(l2[j],20,'\n');
        cout<<l2[j++]<<endl;
        n++;
    }
    m--;
    n--;
    i=0,j=0;
    while(i<m&& j<n)
    {
```

```
        if(strcmp(l1[i],l2[j])==0)
        {
            strcpy(l3[k],l1[i]);
            f3<<l3[k]<<endl;
            i++;
            j++;
            k++;
        }
        else if(strcmp(l1[i],l2[j])<0)
            i++;
        else
            j++;
    }
    if(k==0)
        cout<<"No common names!!\n";
    else
    {
        cout<<"The names that are common are:\n";
        for(i=0;i<k;i++)
            cout<<l3[i]<<endl;
    }
}
```

OUTPUTS:

Enter the number of records in file 1: 4

Enter the names:

Shiva

Puneeth

Rajanikanth

Amithabh

Enter the number of records in file 2: 4

Enter the names:

Sunil

Anil Kapoor

Amithabh

Raj Kapoor

Contents of file 1 are:

Amithabh

Puneeth

Rajanikanth

Shiva

Contents of file 2 are:

Amithabh

Anil Kapoor

Raj Kapoor

Sunil

The names that are common are:

Amithabh

PROGRAM 8:

Write a C++ program to read k lists of names and merge them using 'k way merge' algorithm with k=8

```
#include<stdio.h>
#include<iostream.h>
#include<fstream.h>
#include<string.h>
class record
{
public:
    char name[20],usn[20];
}rec[20];

char fname[8][8]={"1.txt","2.txt","3.txt","4.txt","5.txt","6.txt","7.txt","8.txt"};
fstream file[8];
int n;
void merge_file(char *fil1,char *fil2,char *fil)
{
    record rcd[20];
    fstream f1(fil1,ios::in);
    fstream f2(fil2,ios::in);
    fstream f3(fil,ios::out);
    int i,j,k=0,k1;
    record temp;
    while(!f1.eof())
    {
        f1.getline(rcd[k].name,20,'|');
        f1.getline(rcd[k++].usn,20);
    }
    while(!f2.eof())
    {

```

```
f2.getline(rcd[k].name,20,'|');
f2.getline(rcd[k++].usn,20);
}
k1=k;
for(i=0;i<k-2;i++)
{
    for(j=0;j<k-i-2;j++)
    {
        if(strcmp(rcd[j].name,rcd[j+1].name)>0)
        {
            temp=rcd[j];
            rcd[j]=rcd[j+1];
            rcd[j+1]=temp;
        }
        else if(strcmp(rcd[j].name,rcd[j+1].name)==0)
        {
            if(strcmp(rcd[j].usn,rcd[j+1].usn)>0)
            {
                temp=rcd[j];
                rcd[j]=rcd[j+1];
                rcd[j+1]=temp;
            }
            else if(strcmp(rcd[j].usn,rcd[j+1].usn)==0)
            {
                j++;
                k1--;
                n--;
            }
        }
    }
}
for(i=1;i<k1-1;i++)
{
```

```
        f3<<rcd[i].name<<"|"<<rcd[i].usn<<endl;
    }
    f1.close();
    f2.close();
    f3.close();
}

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

void main()
{
    int i;
    char name[20],usn[20];
    n=0;
    for(i=0;i<8;i++)
        file[i].open(fname[i],ios::out);
    cout<<"Enter number of records: ";
    cin>>n;
    cout<<"Enter Name and USN:\n";
    for(i=0;i<n;i++)
    {
        cout<<"Student "<<i+1<<" : ";
        cin>>rec[i].name>>rec[i].usn;
        file[i%8]<<rec[i].name<<"|"<<rec[i].usn<<endl;
    }
}
```

```
    }  
    for(i=0;i<8;i++)  
        file[i].close();  
    k_merge();  
    fstream output("1111.txt",ios::in);  
    cout<<"\nThe sorted records are:\n";  
    cout<<"NAME\tUSN\n";  
    for(i=0;i<n;i++)  
    {  
        output.getline(name,20,'|');  
        output.getline(usn,20);  
        cout<<name<<"\t"<<usn<<endl;  
    }  
}
```

OUTPUT:

Note: All 8 files are to be populated with the data and the result to be verified in all the intermediate files.

Enter number of records: 8

Enter Name and USN:

Student 1 : BhimsenJoshi 100

Student 2 : Basavaraj 200

Student 3 : MallikarjunMansoor 300

Student 4 : Hariprasadchaurasia 400

Student 5 : ShivakumarSharma 500

Student 6 : ZakirHussain 600

Student 7 : Ravishankar 700

Student 8 : AnushkaRavishnkar 800

The sorted records are:

NAME USN

AnushkaRavishnkar 800

Basavaraj 200

BhimsenJoshi 100

Hariprasadchaurasia 400

MallikarjunMansoor 300

Ravishankar 700

ShivakumarSharma 500

ZakirHussain 600