File Structures Lab Programs

Program1:

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

using namespace std;



/*****************************************************************************
* Function                : main
* Input parameters    : int argc - no of commamd line arguments
*                       char **argv - vector to store command line argumennts
* Returns                 :   0 on success
*****************************************************************************/
int main()
{
        char s1[25];
        fstream f1,f2;
        int i = 0, j = 0, x = 0, c = 0, kb = 0;
        char fname1[25],fname2[25];
        cout << "1:For std i/o\n2.For file i/o\n";
        cout << "Enter your choice :\n";
        cin >> kb;
        switch(kb)
        {
                case 1:
       cout << "Enter number of names :";
       cin >> c;
```

```
                    for(j = 1;j <= c; j++)
                    {
                            cout << "Enter name :" << j << ":";
                            cin >> s1;
                            x = strlen(s1);
                            cout << "Reversed name :" << j << ":";

   // Code to reverse the input
                            for(i = x-1; i >= 0; i--)
                                    cout << s1[i];

   // Display on standard output
                            cout << endl;
                    }
                    break;

            case 2:
cout << "Enter datafile name :";
                    cin >> fname1;
                    cout << "Enter reverse datafile name :";
                    cin >> fname2;

// Opening requested files
                    f1.open(fname1, ios::in);
                    f2.open(fname2, ios::out);

if(!f1)
{
   cerr << "File doesnot exist: " << fname1;
   return -1;
```

```
        }
        if(!f2)
        {
           cerr << "File doesnot exist: " << fname2;
           return -1;
        }
                        while(1)
                        {
                                f1.getline(s1,25);
                                if(f1.fail())
                                        break;
                                x = strlen(s1);
                                for(i = x-1; i >= 0; i--)
             f2 << s1[i];
                                f2 << endl;
                        }
        f1.close();
        f2.close();
        break;
    }
    return 0;
}
```

Program 2:

```
#include<iostream>
#include<string>
#include<fstream>
#include<stdlib.h>                  //stlib is a C library, not a C++ library... thus the ".h"
using namespace std;

class student
{
        public:
                string usn, name, sem;
                void enter_data();      //input name usn sem from the user and stores in object
                void display_data();    //displays an object(name,usn, sem) to the user
                void pack();            //converts an object to a string and writes the string to the
record file
                void unpack();          //reads 1 line in the record file to a string and converts this
string to an object
                void modify();          //accepts new values for name usn and sem from the user
}s[100],temp;
fstream fp;

void search();                          //unpacks all records and searches for particular record... if
it is modified... all the records are again packed
void error(int);                    //displays suitable error messages

int main()
{
        int choice;
        system("clear");            // == clrscr();
```

```
	while(1)
	{
		cout<<"\n1. Insert a Record\n2. Search and modify a record\n3. Exit\nEnter choice: "<<endl;
		cin>>choice;
		switch(choice)
		{
			case 1:
				temp.enter_data();
				fp.open("in.txt",ios::out|ios::app);		//ios::out -> open file for writing, ios::app -> append i.e. write at the end of the file, (existing matter is not overwritten)
				if(!fp)					//check if file has opened successfully, else display 1st error
					error(1);
				temp.pack();
				fp.close();
				break;
			case 2:
				search();
				break;
				default: exit(0);
		}
	}
}
void student::enter_data()
{
	cout<<"\nEnter usn: ";
	cin>>usn;
	cout<<"\nEnter name: ";
	cin>>name;
	cout<<"\nEnter sem: ";
```

```
        cin>>sem;
}

void student::pack()
{
        string buf = usn + "|" + name + "|" + sem + "|";
        if(buf.length() > 45)                   //if rec length> 45 then we have to reject the record by displaying 2nde error
        {       error(2);
                return;
        }
        while(buf.length()<46)                  //if rec len < 45.. extra bytes is padded
                buf += '_';
        fp<<buf<<endl;
}
void search()
{
        string usn_srch;
        int i=0, srch_flag=-1, modify_flag=-1, count;
        cout<<"\nEnter the USN of the student to be found: ";
        cin>>usn_srch;
        fp.open("in.txt",ios::in);
        if(!fp)
                error(1);
        while(fp)                               //unpack all the records in the file and store in objects
        {
                s[i].unpack();
                i++;
        }
        fp.close();
```

```
        count = i;
        for(i=0;i<count;i++)                    //search each of the objects for required usn
        {
                if(s[i].usn==usn_srch)
                {
                        srch_flag=i;
                        break;
                }
        }
        if(srch_flag==-1)
        {
                cout<<"Record Not found!";
                return;
        }
        cout<<"\nRecord Found!\n";
        s[srch_flag].display_data();
        cout<<"\nDo you wish to modify the Record??\n Press 1. to modify, 0. to Cancel\n";
        cin>>modify_flag;
        if(modify_flag)
        {       s[srch_flag].modify();
                fp.open("in.txt",ios::out);
                if(!fp)
                        error(1);
                for(i=0;i<count;i++)
                        s[i].pack();
                fp.close();
        }

}
```

```
void student::unpack()
{
		string seg;
		getline(fp,usn,'|');
		getline(fp,name,'|');
		getline(fp,sem,'|');
		getline(fp,seg);
}

void student::display_data()
{
	cout<<"\nName: "<<name<<"\nusn: "<<usn<<"\nsem: "<<sem<<endl;
}
void student::modify()
{
	int choice;
	while(1)
	{
		cout<<"\nEnter the field to modify:\n 1. name\n 2.usn \n 3.sem \n 4. to exit modification\n";
		cin>>choice;
		switch(choice)
		{
			case 1:  cout<<"\nEnter new name: ";
				cin>>name;
				break;
			case 2:  cout<<"\nEnter new usn: ";
				cin>>usn;
				break;
			case 3: cout<<"\nEnter new sem: ";
```

```
                                        cin>>sem;
                                        break;
                                default: return;
                        }
                }
}

void error(int error_type)
{
        switch(error_type)
        {
                case 1: cout<<"\nFATAL ERROR!: Unable to open the record File\n";
                        exit(0);
                case 2:  cout<<"\n ERROR!: Data exceeds record length\n";
                        return;
        }
}
```

Program 3:

```
#include<iostream>
#include<string>
#include<fstream>
#include<stdlib.h>
using namespace std;

class student
{
        public:
                string usn, name, sem;
                void enter_data();              //NOTE: all functions except pack() are exactly the same.
                void display_data();
                void pack();
                void unpack();
                void modify();
}s[100];
fstream fp;
void search();
void error(int);

int main()
{
        int choice;

        system("clear");

        while(1)
```

```
	{
		cout<<"\n1. Insert a Record\n2. Search and modify a record\n3. Exit\nEnter choice: "<<endl;
		cin>>choice;
		switch(choice)
		{
			case 1:
				s[0].enter_data();
				fp.open("in2.txt",ios::out|ios::app);
				if(!fp)
					error(1);
				s[0].pack();
				fp.close();
				break;
			case 2:
				search();
				break;
				default: exit(0);
		}
	}
}

void student::enter_data()
{
	cout<<"\nEnter usn: ";
	cin>>usn;
	cout<<"\nEnter name: ";
	cin>>name;
	cout<<"\nEnter sem: ";
	cin>>sem;
```

```
}

void student::pack()
{
        string buf = usn + "|" + name + "|" + sem + "|";  //no need of padding and rejecting
        fp<<buf<<endl;
}
void search()
{
        string usn_srch;
        int i=0, srch_flag=-1, modify_flag=-1, count;
        cout<<"\nEnter the USN of the student to be found: ";
        cin>>usn_srch;
        fp.open("in2.txt",ios::in);
        if(!fp)
                error(1);
        while(fp)
        {
                s[i].unpack();
                i++;
        }
        fp.close();
        count = i-1;
        for(i=0;i<=count;i++)
        {
                if(s[i].usn==usn_srch)
                {
                        srch_flag=i;
                        break;
                }
```

```
	}
	if(srch_flag==-1)
	{
		cout<<"\nRecord Not found!\n";
		return;
	}
	cout<<"\nRecord Found!\n";
	s[srch_flag].display_data();
	cout<<"\nDo you wish to modify the Record??\n Press 1. to modify, 0. to Cancel\n";
	cin>>modify_flag;
	if(modify_flag)
	{	s[srch_flag].modify();
		fp.open("in2.txt",ios::out);
		if(!fp)
			error(1);
		for(i=0;i<count;i++)
			s[i].pack();
		fp.close();
	}

}

void student::unpack()
{
	string seg;
	getline(fp,usn,'|');
	getline(fp,name,'|');
	getline(fp,sem,'|');
	getline(fp,seg);
```

```
}


void student::display_data()
{
        cout<<"\nName: "<<name<<"\nusn: "<<usn<<"\nsem: "<<sem<<endl;
}
void student::modify()
{
        int choice;
        while(1)
        {
                cout<<"\nEnter the field to modify:\n 1. name\n 2.usn \n 3.sem \n 4. to exit
modification\n";
                cin>>choice;
                switch(choice)
                {
                        case 1:  cout<<"\nEnter new name: ";
                                 cin>>name;
                                 break;
                        case 2:  cout<<"\nEnter new usn: ";
                                 cin>>usn;
                                 break;
                        case 3: cout<<"\nEnter new sem: ";
                                 cin>>sem;
                                 break;
                        default: return;
                }
        }
}
```

```
void error(int error_type)
{
	switch(error_type)
	{
		case 1: cout<<"\nFATAL ERROR!: Unable to open the record File\n";
			exit(0);
	}
}
```

Program 4:

```
#include<iostream>
#include<string>
#include<fstream>
#include<stdlib.h>
using namespace std;

class student
{
string usn,name,sem;
public: void enter_data();
        void display_data();
        void pack();
        void unpack(int);
}s1;

int rrn[100],cnt=0;
fstream fp1;
void find_rrn();
void search();
void error(int);
int main()
{
         int choice;
         //system("clear");
         fp1.open("rrnrec.txt",ios::out|ios::app);
         fp1.close();
         find_rrn();
         while(1)
```

```
        {
                cout<<"1. Insert  2.Search using RRN 3.Exit \n Enter ur  Choice:- \n";
                cin>>choice;
                switch(choice)
                {
                        case 1:
                            s1.enter_data();
                            fp1.open("rrnrec.txt",ios::out|ios::app);
                            if(!fp1)
                            error(1);
                            s1.pack();
                            fp1.close();
                            break;

                        case 2: search();
                                break;
                    default:exit(0);
                }
            }
        }

void find_rrn()
{
  int pos;
  string seg;
  fp1.open("rrnrec.txt",ios::in);

   if(!fp1)
   error(1);
     while(fp1)
```

```
            {
              pos=fp1.tellg();
              getline(fp1,seg);
              if(seg.length()==0)
                continue;
              rrn[++cnt]=pos;
            }
     fp1.close();
}

void student::enter_data()
{
            cout<<"\n Enter USN:";
            cin>>usn;
            cout<<"Enter Name:";
            cin>>name;
            cout<<"Enter Sem: ";
            cin>>sem;
  }

void student::pack()
{
  int pos=fp1.tellg();
  string buf=usn+"|"+name+"|"+sem+"|" ;
  fp1<<buf<<endl;
  rrn[++cnt]=pos;
}

void search()
{
```

```
 int rrn_srch,pos;
 cout<<"\n Enter RRN of REcord to be found";
 cin>>rrn_srch;
 if(rrn_srch>cnt||rrn_srch<1)
 {
        error(2);
        return;
 }
 cout<<"Record Found";
 pos=rrn[rrn_srch];
 fp1.open("rrnrec.txt",ios::in);
 if(!fp1)
 error(1);
 s1.unpack(pos);
 fp1.close();
 s1.display_data();
}

void student::unpack(int pos)
{
 fp1.seekg(pos,ios::beg);
 getline(fp1,usn,'|');
 getline(fp1,name,'|');
 getline(fp1,sem,'|');
}




void student::display_data()
{
```

```
            cout<<"USN="<<usn<<"Name="<<name<<"Sem="<<sem<<endl;
}


void error(int error_type)
{
  switch(error_type)
   {
     case 1: cout<<"Unable to open file";
           exit(0);
     case 2: cout<<"Invalid RRN";
           return;
 }
}
```

Program 5:

```
#include<iostream>
#include<string>
#include<fstream>
#include<stdlib.h>
using namespace std;
class student
{
        public:
                string usn,name,sem;
                void enter_data();
                void pack();
}s1;

struct index
{
        string usn;
        int addr;
}i1[100],temp;

int cnt;
fstream fp;

void create_index();
void sort_index();
void search();
int bin_srch(string);
void del();
void error(int);
```

```
int main()
{
        int choice;
        //clrscr();
        fp.open("record5.txt",ios::out|ios::app);
        fp.close();
        create_index();

        while(1)
        {
           cout<<"1. Add Record\n 2.Search Record\n 3. Delete Record \n 4.Exit";
           cout<<"Enter Choice: ";
           cin>>choice;
         switch(choice)
         {
          case 1: s1.enter_data();
                fp.open("record5.txt",ios::out|ios::app);
                    if(!fp)
                     error(1);
                    s1.pack();
           fp.close();
                    break;
          case 2: search();
                break;
          case 3: del();
                break;
       default: exit(0);
     }
   }
}
```

```
void create_index()
{
            int pos,i;
            string seg,usnbuf;
            cnt = -1;
            fp.open("record5.txt",ios::in);
            if(!fp)
             error(1);
            while(fp)
            {
               usnbuf.erase();
               pos=fp.tellg();
          getline(fp,usnbuf,'|');
          getline(fp,seg);
          if(usnbuf[0]=='*'||usnbuf.length()==0)
           continue;
          cnt++;
          i1[cnt].usn=usnbuf;
          i1[cnt].addr=pos;
      }
      fp.close();
      sort_index();
}


void sort_index()
{
            for(int i=0;i<=cnt;i++)
            {
```

```
            for(int j=i+1;j<=cnt;j++)
        {

         if(i1[i].usn>i1[j].usn)
         {
                temp.usn=i1[i].usn;
           temp.addr=i1[i].addr;
           i1[i].usn=i1[j].usn;
           i1[i].addr=i1[j].addr;
           i1[j].usn=temp.usn;
           i1[j].addr=temp.addr;
         }
      }
}
}

void student::enter_data()
{
            cout<<"\nEnter USN: ";
            cin>>usn;
            cout<<"\n Enter Name: ";
            cin>>name;
            cout<<"\n Enter Sem:";
            cin>>sem;
}

void student::pack()
{

            int pos=fp.tellg();
```

```
        string buf=usn +"|" + name +"|" + sem +"|";
        fp<<buf<<endl;
        cnt++;
        i1[cnt].usn=usn;
        i1[cnt].addr=pos;
        sort_index();
}

int bin_srch(string usn_srch)
{
        int l=0,h=cnt,mid;
        while(l<=h)
        {
          mid=(l+h)/2;
          if(i1[mid].usn==usn_srch)
      {
        return mid;
      }
        if(i1[mid].usn<usn_srch)
        l=mid+1;
        if(i1[mid].usn>usn_srch)
        h=mid-1;
    }
    return -1;
}

void search()
{
        string usn_srch,buf;
        cout<<"Enter the USN of the student to be found: ";
```

```
        cin>>usn_srch;
        int pos=bin_srch(usn_srch);
        if(pos==-1)
        {
            cout<<"Record Not found \n";
            return;
      }
        cout<<"record found\n";
        cout<<"Usn|Name|Sem"<<endl;
        fp.open("record5.txt",ios::in);
        if(!fp)
          error(1);
        fp.seekg(i1[pos].addr,ios::beg);
        getline(fp,buf);
        fp.close();
        cout<<buf<<endl;
}

void del()
{
        string usn_srch;
        cout<<"Enter the USN to be deleted: ";
        cin>>usn_srch;
        int pos=bin_srch(usn_srch);
        if(pos==-1)
        {
          cout<<"\n Record Not Found \n";
          return;
        }
        cout<<"\n Record found and deleted \n";
```

```
        fp.open("record5.txt",ios::out|ios::in);
        fp.seekp(i1[pos].addr,ios::beg);
        fp.put('*');
        fp.close();
        for(int i=pos;i<cnt;i++)
        {
          i1[i].usn=i1[i+1].usn;
          i1[i].addr=i1[i+1].addr;
        }
        cnt--;
}

void error(int error_type)
{
   switch(error_type)
    {
        case 1: cout<<"\n Unable to open record file \n";
        exit(0);
    }
}
```

Program 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.

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

using namespace std;

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

char st_no[5] , rt_name[20];
int no;
void sort()
{
   int i, j ;
   record temp;
   for(i = 0; i < no-1; i++)
   {
      for( j = 0; j < no-i-1; j++)
      {
         if(strcmp(rec[j].name , rec[j+1].name) > 0)
         {
```

```
            temp = rec[j];
            rec[j] = rec[j+1];
            rec[j+1] = temp;
          }
       }

   }

}

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

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

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

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

    }
    f1.close();
```

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

   }
   no=i;
   f1.close();
   int k=-1;
   for(i=0;i<no;i++)
   {
      if(strcmp(index[i],idx)==0)
      {
         k=i;
```

```
            break;
        }
    }
    if(k>-1)
    {
        for(i=k;i<no-1;i++)
        {
            rec[i]=rec[i+1];
        }
        no--;
        sort();
        create_index_file();
        cout<<"\nData Successfully Deleted\n";

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

}

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

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

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

void retrieve_details(int ch)
{
    int k=0, i;
    char buff[80] , *p;
    ifstream f1;
    char chusn[20] , index[20][80];
    f1.open("secindex.txt" , ios::in);
    while(!f1.eof())
    {
       f1.getline(buff , 80 , '\n');
       p = strtok(buff , "|");
```

```
    if(strcmp(rt_name , p) == 0)
    {
        strcpy(found[k].name , p);
        p = strtok(NULL , "|");
        strcpy(found[k].usn , p);
        p = strtok(NULL , "|");
        strcpy(index[k] , p);
        k++;
    }
}

if(k == 1)
{
    if(ch == 2)
        retrieve_record(index[0]);
    else
        delete_record(index[0]);
}
else if(k > 1)
{
    cout<<"Please choose the candidate USN\n";
    for( i = 0; i < k; i++)
    {
        cout<<"Name = "<<found[i].name <<"USN = "<<found[i].usn<<endl;
    }
    cin>>chusn;
    for(i=0; i<k ; i++)
    {
        if(strcmp(chusn , found[i].usn) == 0)
        {
```

```
            if(ch == 2)
               retrieve_record(index[i]);
            else
               delete_record(index[i]);
         }
      }

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

}

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

```
                cout<<"\nName\t";
                cin>>rec[i].name;
                cout<<"\nSem\t";
                cin>>rec[i].sem;
            }
            sort();
            create_index_file();
            break;
        case 2:
        case 3: if(ch ==2)
                cout<<"Enter the name to search\t";
            else
                cout<<"Enter the student name to delete\t";
            cin>>rt_name;
            retrieve_details(ch);
            break;
        case 4: display_record();
            break;

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

Program 7:

```
// using Consequential Match based on a single loop. Output the names common to
//both the lists.

#include<iostream>
#include<string>
#include<fstream>
#include<ctype.h>
using namespace std;



class conseq
{
	public:
		string list1[100],list2[100];
		int c1,c2;
		void l_list();
		void s_list();
		void match();
};
void conseq::l_list()
{
	fstream fp;
	char name[100];
	c1=-1;c2=-1;
	fp.open("a1.txt",ios::in);
	while(fp)
	{
		fp.getline(name,100,'\n');
```

```
                list1[++c1]=name;
        }
        fp.close();
        fp.open("a2.txt",ios::in);
        while(fp)
        {
                fp.getline(name,100,'\n');
                list2[++c2]=name;
        }
        fp.close();
}
void conseq::s_list()
{
        int i,j;
        string temp;
        for(i=0;i<=c1;i++)
        {
                for(j=i+1;j<=c1;j++)
                {
                        if(list1[i]>list1[j])
                        {
                                temp=list1[i];
                                list1[i]=list1[j];
                                list1[j]=temp;
                        }
                }
        }
        for(i=0;i<=c2;i++)
        {
                for(j=i+1;j<=c2;j++)
```

```
                {
                        if(list2[i]>list2[j])
                        {
                                temp=list2[i];
                                list2[i]=list2[j];
                                list2[j]=temp;
                        }
                }
        }
}
void conseq::match()
{
        int i=0,j=0;
        while(i<=c1&&j<=c2)
        {
                if(list1[i]==list2[j])
                {
                        cout<<"\n"<<list1[i];
                        i++;
                        j++;
                }
                if(list1[i]<list2[j])
                        i++;
                if(list1[i]>list2[j])
                        j++;
        }
}
int main()
{
        conseq c;
```

```
        c.l_list();
        c.s_list();
        c.match();
        return 0;
}
```

Program 8:

```
#include<iostream>
#include<string>
#include<fstream>
#include<stdlib.h>
using namespace std;

class coseq
{
public:
string list[8][50];
string outlist[200];
int count1[8],count2[8];

void read_file(int i);
void sort_list(int i);
void kwaymerge();
};

void error(int);

int main()
{
//system("clear");
coseq c;
for(int i=0;i<4;i++)  // for each of the k lists read_file and sortList.
{                             // for k=8 use(i=0;i<8;i++)
c.count1[i]=0;
c.read_file(i);
c.sort_list(i);
```

```
}
c.kwaymerge();
return 0;
}
void coseq::read_file(int i)
{
fstream fp;
string name;
switch(i)
{
case 0:fp.open("n1.txt",ios::in);
        break;
case 1:fp.open("n2.txt",ios::in);
        break;
case 2:fp.open("n3.txt",ios::in);
        break;
case 3:fp.open("n4.txt",ios::in);
        break;
}
if(!fp)
error(1);
while(fp)
{
getline(fp,name);
if(name.length()>0)
list[i][count1[i]++]=name;
}

fp.close();
}
```

```
void coseq::sort_list(int k)
{
int i,j;
string temp;
for(i=0;i<count1[k];i++)
{
for(j=i+1;j<count1[k];j++)
{
if(list[k][i]>list[k][j])
{
temp=list[k][i];
list[k][i]=list[k][j];
list[k][j]=temp;
}
}
}
}

void coseq::kwaymerge()
{
string sml;
int s_list,count3=0,strt=0,avail_list=4,avail[4],i;
for(i=0;i<4;i++)
{
avail[i]=1;
count2[i]=0;
}
while(avail_list>1)
{
if(!avail[strt]) //if the list is not available
```

```
{
strt++;
continue;
}
s_list=strt;
sml=list[strt][count2[strt]];
for(i=strt+1;i<4;i++)
{
if(!avail[i])
continue;
if(list[i][count2[i]]<sml)
{
sml=list[i][count2[i]];
s_list=i;
}
}
count2[s_list]++;
if(count2[s_list]==count1[s_list])
{
avail[s_list]=0;
avail_list--;
}
outlist[count3++]=sml;
}
for(i=0;i<4;i++)

if(avail[i])
{
for(int j=count2[i];j<count1[i];j++)
outlist[count3++]=list[i][j];
```

```
}
cout<<"\n Merged list:\n";
for(i=0;i<count3;i++)
{
if(outlist[i]==outlist[i+1])
continue;
cout<<outlist[i]<<endl;
}
}
void error(int error_type)
{
switch(error_type)
{
case 1:cout<<"\n FATAL ERROR\n";
      exit(0);
}
}
```