FS LAB PROGRAMS

PROGRAM 2

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

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

void search();

class Student
{
        public:
                string urn;
                string name;
                string sem;
                string Buf;
                char buf[46];
                int pack();
                void write_f(fstream &);
                void unpack();
                void print(ostream &);
                void read_f(fstream &);
};

int Student::pack()
{
```

```
	Buf=urn+"|"+name+"|"+sem+"|";
	if(Buf.length()>45)
		return 0;
	while(Buf.length()<45)
		Buf+="_";
	Buf+="\0";
	return 1;
}

void Student::write_f(fstream &fp)
{
	fp.flush();
	fp<<Buf<<'\n';
	fp.flush();
}

void Student::print(ostream & stream)
{
	stream<<"Student"<<"\t URN"<<urn<<"\n"<<"\t Name:"<<name<<"\n\t Sem:"<<sem;
}
void Student::unpack()
{
	char stg[100];
	int pos=0,count=0,k;
	while(count<3)
	{
		k=0;
		for(int i=pos;i<strlen(buf);i++,k++)
		{
			stg[k]=buf[i];
			pos++;
			if(buf[i]=='|')
```

```
                                        break;
                        }
                        stg[k]='\0';
                        count++;
                        if(count==1)urn=stg;
                        if(count==2)name=stg;
                        if(count==3)sem=stg;
                }
}

void Student:: read_f(fstream &fp)
{
                char sg[55];
                fp.getline(buf,46,'_');
                fp.getline(sg,50,'\n');
}

int main()
{
                int ch,x;
                Student s;
                fstream fp;
                do
                {
                        cout<<"Enter your choice\n";
                        cout<<"1.Insert\n2.Search and Modify\n3.Exit\n";
                        cin>>ch;
                        switch(ch)
                        {
                                case 1:  fp.open("in.txt",ios::out | ios::app);
                                        cout<<"Enter URN => ";
                                        cin>>s.urn;
                                        cout<<"Enter name => ";
```

```
                                        cin>>s.name;
                                        cout<<"Enter Sem => ";
                                        cin>>s.sem;
                                        int k ;
                                        k = s.pack();
                                        if(k==0)
                                        {
                                                cout<<"Invalid data\n";
                                        }
                                        else
                                                s.write_f(fp);
                                        fp.close();
                                        break;
                                case 2:  search();
                                        break;
                                case 3: exit(1);
                        }
                }
                while(ch<=3);

                return 0;
}

void search()
{
                int c=0,choice;
                string usn;
                Student s[1100];
                fstream fp1;
                fp1.open("in.txt",ios::in);
                cout<<"Enter the URN to be searched => ";
                cin>>usn;
                int cnt=0;
```

```
int i=0;
while(fp1)
{
        s[i].read_f(fp1);
        s[i].unpack();
        i++;
}
fp1.close();
cnt=i-1;
for(i=0;i<cnt;i++)
{
        if(s[i].urn==usn)
        {
                c++;
                break;
        }
}
if(c==0)
{
        cout<<"Record not found\n";
        return;
}
else
{
        cout<<"Record found\n";
        s[i].print(cout);
        do
        {
                cout<<"Enter your choice\n";
                cout<<"URN => "<<s[i].urn<<"\n";
                cout<<"1.Name => "<<s[i].name<<"\n2.Sem => "<<s[i].sem<<"\n3.Exit\n";
                cout<<"Enter your choice(1,2,3)";
                cin>>choice;
```

```
                        switch(choice)
                        {
                                case 1:   cout<<"Enter new name => ";
                                          cin>>s[i].name;
                                          break;
                                case 2:   cout<<"Enter new sem => ";
                                          cin>>s[i].sem;
                                          break;
                                case 3: break;
                                default: cout<<"Wrong choice, please enter a valid choice\n";
                        }
                }
                while(choice!=3);
                fp1.open("in.txt",ios::out);
                for(i=0;i<cnt;i++)
                {
                        s[i].pack();
                        s[i].write_f(fp1);
                }
                fp1.close();
        }
}
```

PROGRAM 3

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

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

```
using namespace std;
class student
{
	public:
		string urn;
		string name;
		string sem;
		string Buf;
		char buf[100];
		void pack();
		void write_f(fstream &);
		void unpack();
		void print(ostream &);
		void read_f(fstream &);
};
void student::pack()
{
Buf=urn+"|"+name+"|"+sem+"\n";
}
void student::write_f(fstream &fp)
{
	fp<<Buf;
}
void student::print(ostream &stream)
{
	stream<<"student:"
	<<"\t urn'"<<urn<<"'\n"
		<<"\t name'"<<name<<"'\n"
		<<"\t sem'"<<sem<<"'\n";
}
void student::unpack()
{
	char stg[100];
```

```
		int pos=0,count=0,k;
		while(count<3)
		{
			k=0;
			for(int i=pos;i<strlen(buf);i++,k++)
			{
				stg[k]=buf[i];
				pos++;
				if(buf[i]=='|')
					break;
			}

			stg[k]='\0';
			count++;
			if(count==1) urn=stg;
			if(count==2) name=stg;
			if(count==3) sem=stg;
		}
}
void student::read_f(fstream &fp)
{
		fp.getline(buf,100,'\n');
}
int main()
{
		int ch;
		fstream fp;
		void search();
		student s;
		system("clear");
		do
		{
			cout<<"enter your choice\n";
```

```
                cout<<"1.insert a record\n"
                        <<"2.search and modify a record\n"
                        <<"3.exit\n";
                cin>>ch;
                switch(ch)
                {
                        case 1:fp.open("in.txt",ios::out|ios::app);
                                cout<<"enter urn";
                                cin>>s.urn;
                                cout<<"enter name";
                                cin>>s.name;
                                cout<<"enter sem";
                                cin>>s.sem;
                                s.pack();
                                s.write_f(fp);
                                fp.close();
                                break;
                        case 2:search();
                                break;
                        case 3:exit(1);
                }
        }
        while(ch<=3);
}
void search()
{
        int c=0,choice;
        string usn;
        student s[100];
        fstream fp1;
        fp1.open("in.txt",ios::in);
        cout<<"enter the usn of the student to be searched and modified\n";
        cin>>usn;
```

```
int cnt=0;
int i=0;
while(fp1)
{
        s[i].read_f(fp1);
        s[i].unpack();
        i++;
}
fp1.close();
cnt=i-1;
for(i=0;i<cnt;i++)
{
        if(s[i].urn==usn)
        {
                c++;
                break;
        }
}
if(c==0)
{
        cout<<"record not found\n";
        return;
}
else
{
        cout<<"record found\n";
        s[i].print(cout);
        do
        {
                cout<<"\n\t enter your choice of field to  be modified";
                cout<<"\n\n\t urn=>\t"<<s[i].urn
    <<"\n\n\t 1.name=>\t"<<s[i].name
                <<"\n\n\t 2.semester=>\t"<<s[i].sem
```

```
                    <<"\n\n\t3.exit";
                        cout<<"\n\n\t choice=>";
                        cin>>choice;
                        switch(choice)
                        {
                                case 1:cout<<"enter the name=>";
                                        cin>>s[i].name;
                                        break;
                                case 2:cout<<"enter the semester=>";
                                        cin>>s[i].sem;
                                        break;
                                case 3:break;
                                default:cout<<"\n\t\t\t invalid entry!"<<endl;
                                        break;
                        }
                }
                while(choice!=3);
                fp1.open("in.txt",ios::out);
                for(i=0;i<cnt;i++)
                {
                        s[i].pack();
                        s[i].write_f(fp1);
                }
                fp1.close();
        }
}
```

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>
#include<string>
#include<fstream>
#include<stdlib.h>
#include <cstring>

using namespace std;
char st_no[5];
int no;
class record
{
	public:
		char usn[20];
		char name[20];
		char sem[2];
}
rec[20];
void retrieve_details()
{
	fstream file2;
	char name[20],usn[20],rrn[5],sem[5];
	file2.open("record.txt",ios::in);
	for(int i=0;i<no;i++)
	{
		file2.getline(rrn,5,'|');
		file2.getline(usn,20,'|');
		file2.getline(name,20,'|');
		file2.getline(sem,5,'\n');
		if(strcmp(rrn,st_no)==0)
		{
			cout<<"\n\n"<<"student details are:";
			cout<<"\n\nusn:"<<usn<<"\nname:"<<name<<"\nsem:"<<sem<<"\n";
		}
```

```
        }
        file2.close();
}
int main()
{
        fstream file1,file2;
        int ch;
        char rt_usn[20],st_rrn[20];
        char ind[2],name[20],sem[2];
        int i,flag,flag1;
        file1.open("index.txt",ios::out);
        file2.open("record.txt",ios::out);
        if(!file1||!file2)
        {
                cout<<"file creation error!\n";
                exit(0);
        }
        for(;;)
        {
                cout<<"\n1:add record"<<"\n2:search record";
                cin>>ch;
                switch(ch)
                {
                        case 1:cout<<"enter the no of students:";
                                cin>>no;
                                cout<<"enter the details:\n";
                                for(i=1;i<=no;i++)
                                {
                                        cout<<"\nname:";
                                        cin>>rec[i].name;
                                        cout<<"usn:";
                                        cin>>rec[i].usn;
                                        cout<<"sem:";
```

```
                cin>>rec[i].sem;
                file1<<rec[i].usn<<"|"<<i<<"\n";

file2<<i<<"|"<<rec[i].usn<<"|"<<rec[i].name<<"|"<<rec[i].sem<<"\n";
            }
            file1.close();
            file2.close();
            break;
        case 2:cout<<"enter rrn whose record is to be displayed:";
            cin>>st_rrn;
            file1.open("index.txt",ios::in);
            if(!file1)
            {
                cout<<"\nerror!\n";
                exit(0);
            }
            flag1=0;
            for(i=0;i<no;i++)
            {
                file1.getline(rt_usn,20,'|');
                file1.getline(st_no,4,'\n');
                if(strcmp(st_rrn,st_no)==0)
                {
                    retrieve_details();
                    flag1=1;
                }
            }
            if(!flag1)
                cout<<"record search failed!\n";
                file1.close();
                break;
        default : cout<<"invalid choice";
            exit(0);
```

```
                                        break;

                        }
                }
}
```

PROGRAM 5

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

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

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

```
{
	int t_addr;
	string t_usn;
	cout<<cnt<<'\n';
	for(int i=0;i<cnt-1;i++)
	{
		for(int j=0;j<cnt-1-i;j++)
		{
			if(usn_list[j]>usn_list[j+1])
			{
				t_usn=usn_list[j];
				usn_list[j]=usn_list[j+1];
				usn_list[j+1]=t_usn;
				t_addr=addr_list[j];
				addr_list[j]=addr_list[j+1];
				addr_list[j+1]=t_addr;
			}
		}
	}
}

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

int search( string key)
{
  int pos=0,adr,l=0,h=cnt,mid,flag=0;
  string buffer;
```

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

void del_rec(string key)
```

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

        }
        else
                cout<<"\n Record not found!";
}
int main()
{
        student s[100];
        string key;
        fstream fp;
        for(;;)
        {
        int ch;
```

```
cout<<"\nenter ur choice \n1.add rec\n2. show index\n3.search\n4. delete\n5. exit";
cin>>ch;
switch(ch)
{
	case 1:
		fp.open("inp.txt", ios::out);
		cout<<"enter how many records\n";
		cin>>n;
		for(int i=0; i<n; i++)
		{
			s[i].get_data();
			s[i].add_rec(fp);
		}
		fp.close();
		break;

	case 2: create_index();
		break;

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

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

	case 5: exit(0);
```

```
                }
        }
        return 0 ;
}
```

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<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

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<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

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

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

class coseq
{
        public:
        string list[4][50];
        string olist[50];
        int c1[4], c2[4];
        void l_list();
        void r_file(int i);
        void s_list(int i);
        void merge();
};
void coseq::r_file(int i)
{
        fstream fp;
        char name[100];
```

```
	switch(i)
	{
		case 1: fp.open("n1.txt", ios::in); break;
		case 2: fp.open("n2.txt", ios::in); break;
		case 3: fp.open("n3.txt", ios::in); break;
	}
	while(!fp.eof()){
		fp.getline(name, 100,'\n');
		list[i][++c1[i]] = name;
	}
	fp.close();
}
void coseq::s_list(int k) {
	int i,j;
	string t;
	for(i = 1; i<=c1[k]; i++)
		for(j = i+1; j<=c1[k];j++)
			if(list[k][i] > list[k][j]) {
				t = list[k][i];
				list[k][i] = list[k][j];
				list[k][j] = t;
			}
}
void coseq::l_list()
{
	for(int i=1; i<=3; i++) {
		c1[i] = 0;
		r_file(i);
		s_list(i);
	}
}
```

```
void coseq::merge()
{
	string sml;
	int s_list,i,j;
	int strt = 1;
	int t = -1;
	int av_list = 3;
	int avail[4];
	for(i=1; i<=3; i++){
		avail[i] = 1;
		c2[i] = 1;
	}
	while(av_list > 1) {
		if(!avail[strt]) {
			strt++;
			continue;
		}
		s_list = strt;
		sml = list[strt][c2[strt]];
		for(i= strt+1; i<=3;i++) {
			if(!avail[i]) continue;
			if(list[i][c2[i]] < sml) {
				sml = list[i][c2[i]];
				s_list=i;
			}
		}
		c2[s_list]++;
		if(c2[s_list]>c1[s_list]) {
			avail[s_list] = 0;
			av_list--;
		}
```

```
                olist[++t] = sml;
                for(j = 1; j<=3;  j++)
                        if(j != s_list)
                                if(list[j][c2[j]] == sml)
                                        c2[j]++;
        }
        for(i=1; i <=3; i++)
                if(avail[i])
                        for(j=c2[i]; j <=c1[i];j++)
                                olist[++t]=list[i][j];
        cout<<"\nMerged list : \n";
        for(i= 0; i <=t;i++){
                if(olist[i]==olist[i+1]) continue;
                cout<<olist[i]<<"\n";
        }
}
int main()
{
        coseq c;
        c.l_list();
        c.merge();
        return 0;
}
```