FILE STRUCTURE LAB MANUAL DONE BY ANURAAG A G

PROGRAM 1

Write a program to read series of name, 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

```
#include<iostream>
#include<stdio.h>
#include<cstdio>
#include<fstream>
#include<curses.h>
#include<iomanip>
#include<stdlib.h>
using namespace std;

void reverse(char *s,char *r)
{
        int j,len=0;
        while(s[len]!='\0')
        len++;
        for(j=len-1;j>=0;j--)
        r[len-j-1]=s[j];
        r[len]='\0';
}

int main()
{
        char name[10][20],rev[10][20],input[20],output[20],str[20],rstr[20];
        int i,n,len;
        fstream ifile,ofile;
        curscr;
        cout<<"enter the number of names to read "<<endl;
        cin>>n;
        cout<<"enter the names"<<endl;
        for(i=0;i<n;i++)
{
        scanf("%s",name[i]);
}
        for(i=0;i<n;i++)
{
        reverse(name[i],rev[i]);
}
        cout<<"the names and its reverese order are"<<endl;
        for(i=0;i<n;i++)
        cout<<name[i]<<setw(25)<<rev[i]<<endl;
        cout<<"enter the filename which contain list of names"<<endl;
        cin>>input;
```

```
	ifile.open(input,ios::in);

	if(!ifile)
{
	cout<<"file doesnot exist";
	getch();
	exit(1);
}
	cout<<"enter the filename to store names in reverse order"<<endl;
	cin>>output;
	ofile.open(output,ios::out);
	if(!ofile)
{
	cout<<"file doesnot exit";
	getch();
	exit(1);
}
	while(!ifile.eof())
{
	ifile.getline(str,20,'\n');
	reverse(str,rstr);
	ofile<<rstr<<endl;
}
	getch();
	return 0;
}
```

PROGRAM 2

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

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

class Student {
	public:
	string usn, name, age, branch, sem;

void read() {
	cout << "Enter the student details\n";
	cin >> usn >> name >> age >> branch >> sem;
	}

void pack() {
	string buffer = usn + "|" + name + "|" + age + "|" + branch + "|" + sem;
	ifile << buffer << endl;
	}

void display() {
```

```
        cout << usn << "\t" << name << "\t" << age << "\t" << branch << "\t" << sem << endl;
            }

int search() {
int flag;
        cout << "Enter the USN to be searched:";
        cin >> usn;
        while (!ifile.eof()) {
        unpack();
        if (usn == this->usn) {
        return ifile.tellg();
}
}
        return -1;
}

void modify(int recpos) {
        ifile.seekp(recpos, ios::beg);
        ifile.put('$');
        ifile.seekp(0, ios::end);
        read();
}
};

int main() {
Student s;
int ch;
for (;;) {
        cout << "1. Read\t2. Display\t3. Search\t4. Modify\t5. Exit" << endl;
        cin >> ch;
switch (ch) {
case 1:
        s.read();
        s.pack();
        break;
case 2:
        s.display();
        break;
case 3:
        int flag = s.search();
        if (flag == -1) {
        cout << "Record not found" << endl;
}       else {
        s.modify(flag);
}
        break;
        default:
        exit(0);
}
}
}
```

PROGRAM 3

/* Write a program to read and write student objects with variable -Length records using any suitable record structures. Implemet pack (), unpack (), modify () and search () methods. */

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

using namespace std;

class Student {
public:
 string usn, name, age, branch, sem;

 void read() {
  cout << "Enter the student details\n";
  cin >> usn >> name >> age >> branch >> sem;
 }

 void pack() {
  string buffer = usn + "|" + name + "|" + age + "|" + branch + "|" + sem;
  ifile << buffer << "#";
 }

 void display() {
  cout << usn << "\t" << name << "\t" << age << "\t" << branch << "\t" << sem << endl;
 }

 int search() {
  int flag;
  cout << "Enter the USN to be searched:";
  cin >> usn;
  while (!ifile.eof()) {
   unpack();
   if (usn == this->usn) {
    return ifile.tellg();
   }
  }
  return -1;
 }

 void modify(int recpos) {
  ifile.seekp(recpos, ios::beg);
  ifile.put('$');
  ifile.seekp(0, ios::end);
  read();
 }
};

int main() {
```

```
 Student s;
 int ch;
 for (;;) {
  cout << "1. Read\t2. Display\t3. Search\t4. Modify\t5. Exit" << endl;
  cin >> ch;
  switch (ch) {
  case 1:
   s.read();
   s.pack();
   break;
  case 2:
   s.display();
   break;
  case 3:
   int flag = s.search();
   if (flag == -1) {
    cout << "Record not found" << endl;
   } else {
    s.modify(flag);
   }
   break;
  default:
   exit(0);
  }
 }
}
```

PROGRAM 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. */

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

using namespace std;

class Student {
public:
 string usn, name, age, branch, sem;

 void read() {
  cout << "Enter the student details\n";
  cin >> usn >> name >> age >> branch >> sem;
 }

 void pack() {
  string buffer = usn + "|" + name + "|" + age + "|" + branch + "|" + sem;
  ifile << buffer << "#";
 }
```

```
  void display() {
    cout << usn << "\t" << name << "\t" << age << "\t" << branch << "\t" << sem << endl;
  }

  int search() {
    int rrn, count = 0;
    char dummy[75];
    cout << "Enter the RRN to be searched:";
    cin >> rrn;
    while (!ifile.eof()) {
      if (count == rrn) {
        cout << "Record found\n";
        unpack();
        cout << "USN:" << usn << "\n" << "NAME:" << name << "\n" << "AGE:" << age;
        cout << "\n" << "BRANCH:" << branch << "\n" << "SEM:" << sem << "\n";
        return 1;
      }
      count++;
      ifile.getline(dummy, 100, '#');
    }
    return -1;
  }
};

int main() {
  Student s;
  int ch;
  for (;;) {
    cout << endl << "1. Read\t2. Display\t3. Search\t4. Exit" << endl;
    cout << "Enter the choice:";
    cin >> ch;
    switch (ch) {
    case 1:
      s.read();
      s.pack();
      break;
    case 2:
      s.display();
      break;
    case 3:
      s.search();
      break;
    default:
      exit(0);
    }
  }
  return 0;
}
```

Program 5

/* Write a 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;

class Student {
public:
 string dusn, name, age, branch, sem;

 void read() {
  cout << "Enter the usn no.\n";
  cin >> dusn;
 }

 void pack() {
  string buffer = dusn + "|" + name + "|" + age + "|" + branch + "|" + sem;
 }

 void unpack() {
  cout << dusn << "\t" << name << "\t" << age << "\t" << branch << "\t" << sem << endl;
 }

 void search() {
  int pos = search(dusn);
  if (pos == -1) {
   cout << "Record Not found\n";
  } else {
   unpack();
  }
 }

 void remove() {
  int pos = search(dusn);
  if (pos == -1) {
   cout << "Usn No. not found\n";
  } else {
   stdfile.seekp(atoi(id[pos].addr), ios::beg);
   stdfile.put('$');
  }
 }

 int search(char* fusn) {
  int low = 0, high = indsize - 1;
  int mid;
  while (low <= high) {
   mid = (low + high) / 2;
```

```
    if (strcmp(fusn, id[mid].iusn) == 0) {
      return mid;
    } else if (strcmp(fusn, id[mid].iusn) > 0) {
      low = mid + 1;
    } else {
      high = mid - 1;
    }
   }
   return -1;
  }
};

int main() {
  Student s;
  int ch;
  char susn[15];
  in.initial();
  for (;;) {
   cout << endl << "1. Read\n2. Display\n3. Search\n4. Delete\n5. Exit" << endl;
   cin >> ch;
   switch (ch) {
   case 1:
    s.read();
    break;
   case 2:
    s.dataDisp();
    break;
   case 3:
    cout << "Enter the USN to be searched\n";
    cin >> susn;
    s.search();
    break;
   case 4:
    cout << "Enter the usn no to delete from the record\n";
    cin >> susn;
    s.remove();
    break;
   default:
    exit(0);
   }
  }
  return 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 <iostream>
#include <fstream>
#include <string>

using namespace std;

class Student {
  public:
    string dusn, name, age, branch, sem;

    void read() {
      cout << "Enter the usn number: ";
      cin >> dusn;
      cout << "Enter the name: ";
      cin >> name;
      cout << "Enter the age: ";
      cin >> age;
      cout << "Enter the branch: ";
      cin >> branch;
      cout << "Enter the sem: ";
      cin >> sem;
    }

    void pack() {
      string buffer = dusn + "|" + name + "|" + age + "|" + branch + "|" + sem;
    }

    friend int search(string);
    friend void remove();
    friend void display();
};

int search(string name) {
  fstream file;
  file.open("index.txt", ios::in);
  int pos = -1;
  while (!file.eof()) {
    string line;
    file >> line;
    int index = line.find("|");
    if (line.substr(0, index) == name) {
      pos = stoi(line.substr(index + 1));
      break;
    }
  }
  file.close();
```

```
  return pos;
}

void remove() {
  string name;
  cout << "Enter the name of the student to be removed: ";
  cin >> name;
  int pos = search(name);
  if (pos == -1) {
    cout << "Student not found.\n";
    return;
  }

  fstream file("data.txt", ios::in | ios::out);
  file.seekp(pos, ios::beg);
  file << "$";
  file.close();
}

void display() {
  fstream file("data.txt");
  while (!file.eof()) {
    Student s;
    s.pack();
    file >> s.dusn >> s.name >> s.age >> s.branch >> s.sem;
    cout << s.dusn << " " << s.name << " " << s.age << " " << s.branch << " " << s.sem << endl;
  }
  file.close();
}

int main() {
  fstream file;
  file.open("index.txt", ios::out);
  file.close();

  Student s;
  s.read();
  s.pack();

  file.open("index.txt", ios::app);
  file << s.dusn << "|" << s.name << endl;
  file.close();

  display();

  remove();

  display();

  return 0;
}
```

Program 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 files */

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

using namespace std;

// Function to match the names in two lists and returns a list of the names common to both lists.
vector<string> match_names(ifstream& list1, ifstream& list2) {
  vector<string> names;
  string s1, s2;
  while (getline(list1, s1) && getline(list2, s2)) {
    if (s1 == s2) {
      names.push_back(s1);
    } else if (s1 < s2) {
      s1 = getline(list1, s1);
    } else {
      s2 = getline(list2, s2);
    }
  }
  return names;
}

int main() {
  ifstream list1("name1.txt");
  ifstream list2("name2.txt");
  vector<string> names = match_names(list1, list2);
  for (const string& name : names) {
    cout << name << endl;
  }
  return 0;
}
```

Program 8

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

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

using namespace std;

int main() {
  fstream files[8], outfile;
```

```
 string items[8], min;
 int count = 0;

 for (int i = 0; i < 8; i++) {
  files[i].open(to_string(i) + ".txt", ios::in);
 }
 outfile.open("merge8.txt", ios::out);

 for (int i = 0; i < 8; i++) {
  if (files[i].eof()) {
   count++;
  } else {
   getline(files[i], items[i]);
  }
 }

 while (count < 8) {
  min = "";
  for (int i = 0; i < 8; i++) {
   if (!files[i].eof() && (min.empty() || items[i] < min)) {
    min = items[i];
   }
  }
  count = 0;
  for (int i = 0; i < 8; i++) {
   if (!files[i].eof() && items[i] == min) {
    getline(files[i], items[i]);
    count++;
   }
  }
  outfile << min << "\n";
 }

 for (int i = 0; i < 8; i++) {
  files[i].close();
 }

 return 0;
}
```