Program 5 - **Schedule**
Due Wed Dec 8th At Class Time
100 Points

Write a C++ program called SCHEDULE.CPP that will produce class schedules for students. The program will produce schedules for each student whose student ID is given in the file QUERY.DAT. Producing a class schedule will require obtaining information from four separate input files:

1. STUDENTS.DAT - Contains ordered listing of student IDs and names.
2. SCHEDULE.DAT - Contains the course IDs for each student ID.
3. COURSES.DAT - Contains each course ID, title, and instructor's last name.
4. QUERY.DAT - Contains each student ID that you are producing a schedule for.

Below is an example of each file. The *<EOF>* is a non-visible character indicating the end of the file.

STUDENTS.DAT

```
123
Joe Harding
456
Sara Bison
789
Laura Armstrong<EOF>
```

SCHEDULE.DAT

```
123
COMP170|BNEW101|HIST101
456
ENG111
789
COMP170|BNEW211|POLS202|ART101|HIST101<EOF>
```

COURSES.DAT

```
ART101
Art Appreciation
Choate
COMP170
Intro to C++
Baird
BNEW101
New Testament
Pryor
BNEW211
Life of Christ
Crenshaw
ENG111
Composition I
Jewell
HIST101
American History
Haynie
POLS202
International Relations
Elrod<EOF>
```

Here are some assumptions you may make about these input files:

- Every student-id in the schedule file will be followed by at least one class.
- The maximum number of courses a student may take is 10.
- The student ID will always be a three-digit number.
- There will never be a blank line in any of the files.
- The last line will always have a carriage return on it.
- All formatting will be done precisely as in the examples.
- Max student name length: 30
- Max course title length: 30
- Max instructor name length: 20
- The course ID is composed of two parts: the department and the course number. Each course ID uniquely identifies each class (there can't be two different classes with the same course ID). The department will be 2-4 characters in length. The course number will always be three digits.

Here is an example QUERY.DAT and the resulting class schedules in RESULTS.TXT that would be produced by your program:

QUERY.DAT

```
789
123<EOF>
```

RESULTS.TXT

```
Schedule for Laura Armstrong
Stu ID: 789

Dept  Crs#  Title                          Instructor
----  ----  -----------------------------  --------------------
COMP  170   Intro to C++                   Baird
BNEW  211   Life of Christ                 Crenshaw
POLS  202   International Relations        Elrod
ART   101   Art Appreciation               Choate
HIST  101   American History               Haynie


Schedule for Joe Harding
Stu ID: 123

Dept  Crs#  Title                          Instructor
----  ----  -----------------------------  --------------------
COMP  170   Intro to C++                   Baird
BNEW  101   New Testament                  Pryor
HIST  101   American History               Haynie


<EOF>
```

The RESULTS.TXT file should be formatted exactly as in the above example. Two blank lines follow every schedule, including the last one.

There is a chance that a student ID given in QUERY.DAT does not exist. In this case, your program should output "`Student XXX can't be found`." where XXX is the student ID. Two blank lines should appear after it just like the schedules.

Note: There is no input from the keyboard or output to the screen in this program. All input and output is done through files. Make sure your program operates with the same file names given in the specifications. All files should reside in the same directory as your executable program.

Grading:

Turn in a printout of your documented program **SCHEDULE.CPP**. Submit your program on EASE.

The code and documentation should adhere to the guidelines you have learned this semester to avoid points taken off. Programs not turned in by Friday Dec 10th will not be accepted.

The following is an algorithm that you will need to follow for producing student schedules.

SCHEDULE.CPP algorithm:

1. Get student ID from QUERY.DAT.
2. Search STUDENTS.DAT for student ID and output name and ID to RESULTS.TXT.
3. Search for student ID's class schedule in SCHEDULE.DAT.
4. For each course ID in the schedule, search for course ID in COURSES.DAT and output course info to RESULTS.TXT.
5. Repeat all steps until EOF is reached in QUERY.DAT.

Example program that demonstrates searching STUDENTS.DAT for a specific ID:

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

using namespace std;

void Example1()
{
  ifstream in_file;
  int id;
  bool found = false;
  int search_id;
  char name[30];

  cout << "Enter search ID: ";
  cin >> search_id;
  cin.get();

  in_file.open("STUDENTS.DAT");

  in_file >> id;
  in_file.get();  // Read \n

 // Read from file until hitting EOF or finding ID
  while (!in_file.eof() && !found)
  {
    in_file.getline(name, 30);

    if (id == search_id)
    {
      found = true;
      cout << "My name is " << name << ".\n";
    }
    else
    {
      in_file >> id;
      in_file.get();  // Read \n
    }
  }

  if (!found)
  {
    cout << "ID " << search_id << " wasn't
found!\n";
  }
}
```

```
void Example2()
{
  const int MAX_ID_LENGHT = 10;

  ifstream in_file;
  char id[MAX_ID_LENGHT];
  bool found = false;
  char search_id[MAX_ID_LENGHT];
  char name[30];

  cout << "Enter search ID: ";
  cin.getline(search_id, MAX_ID_LENGHT);

  in_file.open("STUDENTS.DAT");

  in_file.getline(id, MAX_ID_LENGHT);

 // Read from file until hitting EOF or finding ID
  while (!in_file.eof() && !found)
  {
    in_file.getline(name, 30);

    if (strcmp(id,search_id) == 0)
    {
      found = true;
      cout << "My name is " << name << ".\n";
    }
    else
    {
              in_file.getline(id, MAX_ID_LENGHT);
    }
  }

  if (!found)
  {
    cout << "ID " << search_id << " wasn't
found!\n";
  }
}

void main()
{
       Example1();
       Example2();
}
```