CS 240 Fall 2017: Coding Assignment 5

For this assignment, you must work in groups of 2 or 3. Let your TA know if you have trouble finding a partner. It may be easier to have a partner from your lab, but that is not necessary.

There are programming constraints at the bottom of this document; make sure you read through to the end before you start implementing anything, please.

Introduction

As Binghamton University students and Computer Science majors, you are aware that the faculty and administration of the university place certain constraints on you, in terms of the courses you must take to earn a CS degree. We identify "required" courses and "prerequisites" that must be taken before certain other classes. And we don't always offer every course every semester. Therefore, one of your *unwritten* degree requirements is that you can navigate this set of requirements to produce a schedule that will allow you to graduate "on time." This assignment asks you to write a program that is intended to help students "check" a proposed course plan to see if it meets the CS Program requirements, subject to course offerings.

Your program will take as input three files, all named on the command line, as follows:

```
schedule.exe <Requirements File> <Offerings File> <Planned Schedule File>
```

Requirements File

The Requirements File contains information about the program's degree requirements. In particular:

- Exactly one line somewhere in the file must begin with the string "**TOTAL**" and then be followed by a single integer (after whitespace). This integer represents the number of CS credits necessary to complete the program.
- The file also contains zero or more Additional Credit Requirements lines (or simply "Credit" lines). Each Credit line starts with the string "**CREDIT**", is followed by a single letter, and then a number of credits (both separated by whitespace). Courses in the Offerings File, as described below, will be tagged with the letters to indicate that they may count toward this credit requirement.
- "Required Course and Prerequisite" lines (or simply "Course" lines) begin with "**COURSE**". Course lines indicate the name of the course, which is a two letter sequence (e.g. CS) followed by a three digit number. Next on the line is a single character... M for Mandatory, R for Required, and O for Optional. No other letters are allowed in this position on the line. Finally, each Course line ends with a list of zero or more other courses that are prerequisites for this class. If a course is a prerequisite for another course, you may assume that it is listed in the Requirements File.
- Finally, "Class Choice lines" (or simply "Choice" lines), begin with "**CHOOSE**". Next on the line is an integer indicating how many of the courses on the line need to be taken; this integer is followed by a list of two or more courses. So for example, a line that says "CHOOSE 2 CS428 CS457 CS441" would mean that all valid course plans would have to pick at least two out of those three classes.

Here is a sample Requirements File.

```
TOTAL 53
CREDIT C 8
CREDIT H 20
CREDIT F 14
COURSE CS101 M
COURSE CS110 R
```

```
COURSE CS140 R CS110
COURSE CS210 R CS110
COURSE CS220 R CS210
COURSE CS240 R CS140 CS210
COURSE CS325 R CS220
COURSE CS333 R MA314 CS240
COURSE CS350 R CS240 CS220
COURSE CS373 R MA314
COURSE CS471 R
COURSE CS495 M
COURSE MA221 R
COURSE MA222 R MA221
COURSE MA314 R MA221
COURSE MA327 R MA327
COURSE BI118 O BI117
COURSE CH108 O CH107
COURSE PH132 O PH131
CHOOSE 1 MA304 MA371 MA381
CHOOSE 1 BI118 CH108 PH132
```

In addition to the prerequisites indicated within the Requirements File, all CS courses that are *not* listed in the file should be assumed to have all of the Required courses (i.e. those marked with 'R') as prerequisites. (They do not, however, have Mandatory classes as prerequisites... that's the difference between Mandatory and Required, for the purposes of this assignment.)

Any of these lines may appear in any order within the file. The lines may be listed in the order shown above, or they may be "shuffled" in any way (e.g. a "Choice" line may even be the first line in the file).

Your code only needs to consider program (i.e. CS) requirements, not "Gen. Ed." style requirements.

Course Offerings File

A Course Offerings File very simply has a list of courses offered, along with their number of credits, when they are offered (Spring, Fall, or both), and which "additional credit requirements" they satisfy, if any. Each line of the file is formatted as follows:

```
<Course Name> <Credits> <When Offered> [<Tags>]
```

- `<Course Name>` is a two-character department followed by the 3-digit course number, as per the description in the Requirements File section above.
- `<Credits>` is a number of credits for this class, from 1 to 4.
- `<When Offered>` is S for "Spring only", F for "Fall only", or "E" for "Every semester". (Let's assume there are no courses that are "on the books" but that are never offered!)
- Finally, the `<Tags>` consist of zero or more letters that may correspond to the letters indicated on the Credit lines in the Requirements File. Tags are not separated from one another by whitespace. A course with multiple tags may count toward multiple Additional Credit Requirements.

Here is a sample Course Offerings File:

```
CS101 1 E
CS110 4 E
CS140 4 E
```

```
CS210 4 E
CS220 4 E
CS240 4 E
CS325 4 E
CS333 4 E
CS350 4 E
CS373 4 E
CS471 4 E
CS495 1 E
MA221 4 E
MA222 4 E
MA314 4 E
MA327 4 E
BI117 4 F
BI118 4 S
CH108 4 S
CH107 4 F
PH131 4 F
PH132 4 S
MA304 4 E
MA371 4 E
MA381 4 E
EN125 3 F FHC
EN166 3 S FHC
PS204 3 F FH
HI204 3 F FH
AN301 3 E FHC
<etc.>
```

You may assume that any two-character sequence followed by a 3 digit number is a valid course, and that any course name that does not meet that format is invalid.

Course offerings may appear in any order in the file.

Planned Schedule File

A Planned Schedule File contains one line per semester. The semester is named as the first line of the file.

```
F2016 CS140 MA221 HI101 CS101
S2017 CS210 MA222 BW300 PS200
F2017 CS220 MA304 BI117 PH131
S2018 CS240 MA314 BI118 CW201
F2018 CS333 CS325 MA327 GE322
S2019 CS350 CS373 CS441 MS111
F2019 CS428 CS471 EN400 PY209
S2020 CS433 CS472 CS495 MA444
```

(I filled these in with some made up department names and course numbers.)

Semesters are indicated by a single character ('S' for Spring and 'F' for Fall are the only two allowed... we won't consider summer or winter classes), followed by the year.

Course lines may appear in any order in the file... the "order" of courses taken (for prerequisite purposes) is determined by the "S2010" vs. "F2011" strings on the lines they appear on, for example... not by the order they occur in the file.

Program Operation and Output

Your job in your program, then, is to read in these three files and determine whether the Student's plan in the Planned Schedule File will result in graduation, subject to the constraints of the Course Offerings and Requirements Files. You may assume that the student passes all the courses. Your program can have one of three different kinds of output:

1. "**Good plan. Get to work.**"
2. "**Bad plan. Here's why:**"
   The program should then print at least one descriptive reason why the plan cannot be carried out, or would not result in the student meeting the program requirements. ("Descriptive" output will identify courses specifically, rather than including generic output like "Requirements not satisfied.")

You may assume that input files are well formatted, as described above.

Programming Constraints

For full credit:

- Use a **graph** for the prerequisites, and some kind of graph traversal to check that prerequisites have been taken before a student plans to take each class.
- Use an STL **hash_map** to store and lookup course names (e.g. “CS240”) in constant time. You will have to investigate the online documentation to learn how to use this data structure; the hash table terminology you encounter will be familiar, from class, but you will not have to implement (for example) collision resolution or anything else… you just need to use the provided hash_map interface.

The value of this assignment comes in applying some of the things you have learned this semester (reading files, storing data), and thinking about how to organize and store several different forms of data in a way that allows you to access it as required. Design your C++ classes and containers, with your teammate(s), *before* you start coding.

There *are* several “trivial” aspects of the assignment, assuming your program is designed well. For example, do not overthink how you ensure that enough credits are taken… add up all the credits in the Planned Schedule File to make sure there are enough. On the other hand, figuring out how to ensure that prerequisites are satisfied may require more thinking, and some application of a graph algorithm.