CS 203 F20 Project 1:

C W Liew Version as of: 15:46 Saturday 5th September, 2020

Due: 11:59pm, October 17, 2020

Goals

The goals of this project are to:

1. gain experience in designing and implementing C programs,
2. gain a better understanding of how to implement data structures in C
3. gain a better understanding of data structures and functions

General Description

In this project you will design and implement (in the C programming language) a complex data structure that resembles a graph of dependencies. The program reads in the basic network information and builds the data structure. After that it will (in a loop) read in commands from the console that will allow the user to search and manipulate the data structure. The result of executing each command is printed out to the console upon completion of each command. The command line used to start the program will also have a list of filenames (space separated). The files are used as input to populate the data structures. The file input consists of 2 types of files - department information and degree information. The first line of the file contains a keyword that distinguishes which type of file it is.

- for each department file ⇒ read in dept info and a set of course descriptions
- information about departments is stored in an ArrayList of departments
- for each department ⇒ store information about courses in an ArrayList of courses
- for each course ⇒ contains information about course name, title, and a LinkedList of prerequisites (course names)
- for each degree file ⇒ read in degree name, course requirements limited to (implicit)conjuncts and explicit disjuncts)
- information about degrees is stored in an ArrayList of degrees
- the course requirements are stored in a LinkedList of requirements. Each item in the ArrayList consists of either (1) a single course name, or (2) a list of course names - this is a disjunct; you can choose to keep or omit the keyword 'OR' at the beginning of the line

The data structures you will have to implement (you should have learned most of this in lab) are:

- LinkedList
- ArrayList (ignore deletion/shrink; start with 10 and double each time)

Running The Program

Filenames will be passed on the command line as part of the input to the program. For example, if your program is named *p1* and you have input files with the names *file1* and *file2* the program will be started as:

```
./p1 file1 file2
```

file1 and *file2* are assumed to contain information about a department or a degree (see the next section).

File Format

Each file starts with a single line that specifies where it is a department or degree specification with a single keyword (DEPARTMENT or DEGREE). The format for the rest of the lines for each type of file are listed below:

- department info:

```
name of department e.g., Computer Science
course #1 name e.g., CS 205
course #1 title e.g., Software Engineering
course #1 prereqs e.g., MATH 161, MATH 186
course #2 info ...
```

- degree info:

```
name of degree e.g., BA Computer Science
OR CS 104, CS 105, CS 106
CS 150
OR MATH 186, MATH 286, MATH 336
```

Commands

The commands that can be used are:

- **c**: takes one parameter (course name)

  shows the title and pre-requisites (or NOT FOUND)

```
c CS 205
Software Engineering
CS 150

c CS 208
NOT FOUND
```

- **d**: takes one parameter (degree name)

  shows the course requirements for the degree

```
d BA Computer Science
OR CS 104, CS 105
CS 150
CS 202
CS 203
CS 205
OR CS 301, CS 303, CS 320, CS 420, CS 470
OR MATH 141, MATH 161
MATH 182
OR MATH 186, MATH 286, MATH 336, PSYC 120
```

- **s**: takes one parameter (course name)

  shows effect of taking a course; shows courses that can now be taken (it's a prerequisite) and degrees that require the course

```
s CS 150
CS 202, CS 203, CS 205
BS Computer Science
BA Computer Science
```

- **p**: takes two parameters (type, name)

  prints information associated with name. The type can be c - course, d - department, or g - degree

```
p c CS 205
department: Computer Science
degree: BA Computer Science, BS Computer Science
pre-requisites: CS 150
```

- **x**: has no parameters

  exits the program

Grading

Your program will be graded on the following criteria:

- functionality (60%) - how much of the specified functionality works?
  - basic (30%) - reads the files and stores information about departments and courses; prints information about departments and courses; can exit on command
  - good functionality (15%) : stores information about degrees; prints information about degrees; can handle degree requirements (conjuncts only)
  - very good functionality (10%) : shows effect of taking a course, can handle conjunctive degree requirements
  - max functionality (5%) : shows effect of taking a course, can handle disjunctive degree requirements
- design (20%) - is your program decomposed in an appropriate manner?
- documentation (20%) - how well commented is the program

+ 1/2 pgs of what feats/what works