School of Computing & Information Technology

CSCI251/CSCI851 Advanced Programming Spring 2019

Assignment 1 (Worth 10%)

Due 11:55pm Friday 30^{th} August 2019. (End of Week Five)
MAYBE START of WEEK SIX due to CSCI203/CSCI803.

Overview

This assignment is to be implemented using procedural programming. The overall program should follow the progress of students through a sequence of subjects that are part of an academic course at an educational institute : **ABC** (A Big College).

General code notes

These are some general rules about what you should and shouldn't do.

1. Your assignment should be organised into:
   (a) A driver file containing your `main()` function.
   (b) A header file containing the prototypes for the functions you write.
   (c) An implementation file containing the implementations of your functions.
2. Provide a text file `Readme.txt` with instructions for compiling your code on Banshee into the executable `ABC`. The `Readme.txt` should actually be a text file, not a doc or pdf or rtf or something other than text, and the instruction should be a copy and pastable command.
3. **If your code doesn't compile on Banshee you will likely receive 0 for the assignment.**
4. You are not allowed to use classes.
5. You can use structs, but not member functions in them.
6. Within your code, be consistent in your tabbing style. We want readable code.
7. Include sensible volumes of commenting.

8. Use appropriate variable names.
9. Don't leak memory.
10. Your `main()` function should make it clear what is going on, and shouldn't be too large.
11. Other than the initial command line input, the program should run without user output. In particular this means there shouldn't be pauses waiting for the user to press a key.

Run structure

Once your program is compiled into the executable `ABC`, it must run as follows:

```
$ ./ABC Students.txt Subjects.txt Teachers.txt Output-file
```

The files serve particular purposes and by purpose should be assumed to be in this order. The names and content of the files may differ, so you shouldn't hard code the content of the sample files into your program, or hard code the names of files into your program.

The expected structure of the input data files is given in the next section.

When you read from the data files you should report on the data read in. We should see a list of students, a list of subjects, and a list of teachers; all appropriately formatted so it's clear you have correctly partitioned the data. This report should go to standard out, not to `Output-file`. `Output-file` is used to report on results. It should be clear that you have correctly linked files, that should be clearer once you read the format of the data files.

The students in the students file are to be processed in the order they are given. The run of each student is independent. Each student attempts to go through their plan of study taking each subject in the order listed in their plan. A student cannot take a subject until they completed the previous one on their plan. If a student fails a subject three times in row they are to be excluded from the college.

The file `Output-file` should be ordered by student and contain a clear report on the progression of each student, including subject, teacher, and performance for each time they take a subject.

When you start each student you should report to standard out, with the student name and the name of the program they are studying for. You should also report when the student moves to a new subject or attempt at a subject, and when they finish their study.

The process involved in a student taking a subject is as follows:

- For the given subject randomly allocate a teacher who is allowed to teach the subject.
- Apply the teacher and student modifiers to the subject distribution. The modified mean and standard deviations are determined as follows:

  ```
  Mean = Student ability - Subject difficulty - Teacher toughness
  Standard deviation  = Student consistency + Subject variability + Teacher variability
  ```

  Here goes an example using the data listed in the next section. If `Average Ant` is taking `Basic Bouncing`, and being taught by `Tough Terrapin`, the mean and standard deviations become:

```
Mean = 50 - (-10) - (15) = 45
Standard deviation = 5 + (-2) + (3) = 6
```

Poor `Average Ant` doesn't have a great chance of passing...

At this point you should report the student name, subject attempted, teacher teaching, and the distribution parameters to the `Output-file`.

- Determine the mark obtained by the student by generating a random value drawn from the subject distribution. Values less than 0 or more than 100 should be appropriately modified so the mark is an integer in the range 0 to 100.
- Based on the mark three different actions are possible:
  1. If the mark is in the range 0-44 inclusive, the student is given an F (for fail) grade.
  2. If the mark is in the range 45-49 inclusive, the student is given a supplementary assessment. Generate a new value from the distribution and add +5. If the new mark is in the range 0-49 the student fails and received an F grade. If the new mark is in the range 50-100, the student receives a 50-PS. The supplementary assessment mark should be recorded and reported along with the original mark.
  3. If the mark is in the range 50+, the student is given a grade in accordance with the following table, where the ranges are inclusive of the values at the end of range:

Mark	Grade
50-64	P
65-74	C
75-84	D
85-100	HD

- As noted earlier, if a student fails a subject they need to retake it, with a maximum of three attempts allowed. For a first retake of a subject a modifier of +5 is added to the mark obtained. For the second retake a modified of +10 is added to the mark obtained. In each retake the teacher is independently randomly generated. If no more retakes are allowed, the exclusion of the student should be reported to the `Output-file` and you should move on to the next student.
- The mark and grade for each attempt should be reported to the `Output-file`.
- If the student has passed the subject they proceed to the next subject, if there is one. If the student has completed all subjects, they graduate and this should be reported to the `Output-file`.

Inputs

Three data files are needed for each run of the program. The general syntax of those files is described here, and one example of each is provided on Banshee in `/share/cs-pub/251/Assignments/One/`. Those

files are in Unix format so if you copy them over to Windows and back to Unix there may be additional characters, particularly the end of line ^M that may appear.

The three data files are as follows, with the commas and colons used to separate fields.

1. `Students.txt`: No more than 10 entries.

   ```
   Name,Student code,Ability,Consistency,Program name:Subject list
   ```

   Example:

   ```
   Average Ant,204932,50,5,Short course:1
   Brilliant Bison,234543,80,3,Bachelor of Bounciness:2,5,3
   Consistent Canary,123456,60,1,Diploma of Doggeral:3,6
   Dusty the Dinosaur,000001,65,3,Master of Extinction:1,2,3,4,5,6
   Iggy the Irratic,369523,50,15,Some Degree of Oddness:7,5,3,1
   ```

   The student code is a 6 digit string. Note that it can have leading 0's as in Dusty the Dinosaur.

   The ability is an integer in the range 0 to 100 inclusive and represents the mean mark for the student prior to applying subject or teacher modifiers. The consistency is a value in the range 0 to 15, and is the standard deviation prior to applying the teacher and subject modifiers.

   The subject list contains a list of integers with each number corresponding to a subject listed in the subject file. There won't be more than 10 subjects for any student.

2. `Subjects.txt`: No more than 10 entries.

   ```
   Name,Difficulty,Variability
   ```

   Example:

   ```
   Archery,-15,1
   Basic Bouncing,-10,-2
   Counting for Animals,0,2
   Digging,4,1
   Better Bouncing,0,2
   Finding Friends,10,3
   Gathering Greenery,15,0
   ```

   The name cannot be empty. The difficulty is an integer modifier for the mean in the range of -15 to 15 inclusive. The variability is an integer modifier for the standard deviation in the range of -3 to 3 inclusive.

3. `Teachers.txt`: No more than 10 entries.

   ```
   Name,Toughness,Variability,Subject list
   ```

   Example:

```
Tough Terrapin,15,3:2,3,4
Softy Squid,-15,-3:1,2,3,4,5
Moderate Monkey,0,0:4,5,6,7
Contrary Cat,3,3:3,5,6
```

The name cannot be empty. The toughness is an integer modifier for the mean in the range of -15 to 15 inclusive. The variability is an integer modifier for the standard deviation in the range of -3 to 3 inclusive.

The subject list contains a list of integers with each number corresponding to a subject listed in the subject file. There won't be more than 10 subjects for any teacher. These represent the subjects the teacher is allowed to teach.

Output

If there is a problem with one of the three input data files, such as it doesn't open or it contains invalid data, a report should be made to standard error, and the program should abort. The error should be detailed enough to unambiguously identify the problem.

The output was explained in the `Run structure` section.

Notes on submission

Submission is via Moodle.

Your code must compile on Banshee with the instructions you provide. If it doesn't you will likely be given zero for the programming part of this assignment.

Please submit your source, so .cpp and .h files, and your Readme.txt file directly to Moodle. There shouldn't be other files or directories and they shouldn't be in a zip file.

The `Readme.txt` file should contain your compilation instruction.

1. The deadline is 11:55pm Friday 30^{th} August 2019. (End of Week Five).
2. Late submissions will be marked with a 25% deduction for each day, including days over the weekend.
3. Submissions more than three days late will not be marked, unless an extension has been granted.
4. If you need an extension apply through SOLS, if possible **before** the assignment deadline.
5. Plagiarism is treated seriously. Students involved will likely receive zero.