COMP 2401 -- Assignment #2

Due: Thursday, October 23, 2014 at 9:00 PM

Goal

You will write a program in C, in the Ubuntu Linux environment, to help the user compute their final grade in a course, based on multiple weighting schemes. Your program will prompt the user to enter the major course *components*. These are the items that are given weights in the course outline, for example Assignments, Tutorials, Tests, Final exam, etc. Then your program will prompt the user to enter some *weighting schemes*, with one value for each course component. For example, one weighting scheme could have 50% for Assignments, 10% for Tutorials and 40% for the Final exam. Another weighting scheme could weigh the Assignments at 40% and the Final exam at 60%. Finally, your program will prompt the user to enter the grade they received for each component, and print out the final grade the user would get based on each weighting scheme.

In this assignment, you will:

- define the data type that represents the collection of course components and weighting schemes
- break down the required functionality into modular, reusable functions
- write the functions to prompt the user for the required course component and weighting scheme data, as well as the user's grades, and compute a final grade using each weighting scheme

Learning Objectives

- work with arrays and structures
- perform simple pointer manipulations
- practice pass-by-reference parameter passing

Instructions

1. Data structure

Define a single data type to represent all the course component and weighting scheme information. Course components consist of items defined in a course outline, such as "Assignments", "Tests", "Tutorials", etc. Weighting schemes indicate a weight assigned to each course component, e.g. assignments are worth 35% of the final grade, tests are worth 20%, tutorials 10%, etc. All collections must be represented as arrays, which means that you must track the number of elements in each one.

Your data type must include:

- the course components, represented as an array of strings
- the number of course components
- the weighting schemes, represented as a two-dimensional array
  - each element in the first dimension contains the data for one weighting scheme
  - each element in the second dimension indicates the weight of one course component in that weighting scheme
  - the indexes in the course component array must correspond to the indexes in the second dimension of the weighting scheme array
- the number of weighting schemes

2. Functionality

Write a **`main`** function and supporting functions that implement the required functionality:

- define the weighting schemes variable as a local variable in the **`main`** function
  - there must be only **one** instance of the weighting scheme data in the entire program; do not make copies!
  - this is **not** a global variable
- prompt the user to enter the course component data
- prompt the user to enter the weighting scheme data
- prompt the user to enter their grade for each course component
- compute a final grade using each weighting scheme and the user's grades
- print out the final grades to the screen

Notes:

- your program must be user friendly and use component names in prompts (see sample output on the next page)
- your program must perform all basic error checking (e.g. weights for each scheme must add up to 100)

Constraints

- do **not** use any global variables
- compound data types **must** be passed by reference, not by value
- you must reuse functions everywhere possible
- your program must be thoroughly commented

Submission

You will submit in *cuLearn*, before the due date and time, one **`tar`** file that includes all the following:

- all source code
- a readme file, which must include:
  - a preamble (program author, purpose, list of source/header/data files)
  - exact compilation command
  - launching and operating instructions

Grading

- **Marking breakdown:** The grading scheme is posted in cuLearn.
- **Deductions:**
  - Up to 50 marks for any of the following:
    - the code does not compile using gcc in the VM provided for the course
    - code cannot be tested because it doesn't run
  - Up to 20 marks for any of the following:
    - your program is not broken down into multiple reusable, modular functions
    - your program uses global variables (unless otherwise explicitly permitted)
    - your program passes compound data types by value
    - the readme file is missing or incomplete
  - Up to 10 marks for missing comments or other bad style (indentation, identifier names, etc.)
- **Bonus marks:**
  - Up to 5 extra marks are available for fun and creative additional features

Sample Output

You can view a demo of the program here. Below is the output of a sample execution:

```
Don't Panic ==> a2

How many marking components in the course? 4
  enter next component name: Assignments
  enter next component name: Tutorials
  enter next component name: Tests
  enter next component name: Final

How many marking schemes? 2

Marking scheme #1:
  enter Assignments weight: 30
  enter Tutorials weight: 10
  enter Tests weight: 25
  enter Final weight: 55
values do not add up to 100!  Try again...

Marking scheme #1:
  enter Assignments weight: 30
  enter Tutorials weight: 10
  enter Tests weight: 25
  enter Final weight: 35

Marking scheme #2:
  enter Assignments weight: 30
  enter Tutorials weight: 10
  enter Tests weight: 15
  enter Final weight: 45

Enter your Assignments mark: 86.9
Enter your Tutorials mark: 100
Enter your Tests mark: 62.5
Enter your Final mark: 70

Grade using marking scheme #1:  76.19
Grade using marking scheme #2:  76.94

Don't Panic ==>
```