SCIT-EIS-UOW

CSCI251 Advanced Programming

Spring 2023

Assignment 1 (Worth 8%)

Due date: 11:55pm Friday 1st Sept 2023 (End of week 6)

Overview

This assignment is to be implemented using procedural programming. These are some general rules about what you should and should not do.

1. Your assignment should be organised into one zip file with name following the format "familyname_studentId_ass1.zip". The zip file ONLY contains three source-code files and one README:

(a) A driver file **driver.cpp** containing your main() function. Meanwhile, the main() function should make it clear what is going on, and should not be too large.

(b) A header file **header.h** containing the prototypes for the functions you write.

(c) An implementation file **header.cpp** containing the implementations of your functions.

(d) In README, describe the role and content of each source-code file, and how to run your code.

2. Your code must be C++17-compliant and work on **capa.its.uow.edu.au**.

3. We want readable code with a good code layout.

4. Include sensible volumes of commenting.

5. Use appropriate and self-explained variable/function names.

Description

This program is used to simulate the process of a crowdsourcing system, and we simplify some steps here. The **Tasks.txt** and **Workers.txt** file save the information of individual tasks

and workers, respectively.

Each task (one line record from in **Tasks.txt**) represents a task which needs to be assigned to a worker to finish. The crowdsourcing system attempts to assign the task through a list of workers in the same line in **Tasks.txt**. All workers in the list will try the task **in order**. Each worker can try **five times**. Whether this worker is **successful** depends on the **average performance** over five times.

The next worker can try only when the current worker fails. If the current worker y succeeds, you should output something like "Assignment of Task x to worker y succeeds" where x is the taskID and y is the workerID. Otherwise, output something like "Assignment of Task x to worker y fails".

Now, let us define the inputs in **Tasks.txt** and **Workers.txt, 'worker performance' at a time, 'worker average performance' and the 'successful condition' of a worker.**

Inputs in Tasks.txt and Workers.txt:

1. Tasks.txt

   taskId,description,uncertainty,difficulty,workers:a list of worker IDs

   Example:

   123,image labelling,5,10,workers:0,1

2. Workers.txt

   workerId,name,variability,ability

   Example:

   0,Michael,-2,50

Worker Performance and her/his average performance:

The performance score is a sample drawn from a normal distribution. In this normal distribution, "mean = worker ability – task difficulty" and "standard deviation = task uncertainty +worker variability". E.g, given the task and the worker in the example above. Mean=50-10, and standard deviation=5+(-2). **The average performance** is the average score of five independent draws from this distribution.

Successful condition of a worker

We say the worker is successful if and only if the average performance score is greater than 50. Otherwise (i.e., the score <= 50), it fails.

Output:

The tasks from the "**Tasks.txt**" file are to be processed in the order they are given, and each task is independent. You should output the information of each task (i.e., all attribute information in the input file) being processed and the evaluation results for the workers in the list (i.e., Assignment of Task x to worker y succeeds or not). You should output something (like below) into the "**Output.txt**" file.

```
================================================================
processing taskId:      1234
description     :       knowledge fusion
uncertainty     :       5
difficulty      :       20
workers :       0,3,4

----------------------------------------------------------------
Trial: workers  : 0
----------------------------------------------------------------

The average performance is 34
Assignment of Task 1234 to worker 0 fails

----------------------------------------------------------------
Trial: workers  : 3
----------------------------------------------------------------

The average performance is 52
Assignment of Task 1234 to worker 3 succeeds
```

Note:

1. Do not change Tasks.txt and Workers.txt. Report results for tasks in order.
2. Use structs to store information for tasks and workers.
3. Name your executable file as "run". It should run like:
   $ ./run Tasks.txt Workers.txt Output.txt
4. Using any knowledge not covered in lectures (i.e., week 1 to week 4) and this file will cause score deduction.
5. Any standard input/output libraries (e.g., stringstream) can be used.
6. Vectors are not allowed. If you really need something similar, arrays are allowed and the size can be predefined. The number of workers and tasks in the test will be not greater than 20.
7. Academic misconduct is treated seriously. Any plagiarised work will be given a zero mark and reported to the University.