SCIT-EIS-UOW

CSCI251 Advanced Programming

Spring 2024

Assignment 1 (Worth 8%)

Due date: 11:55pm, August 30, 2024

Overview

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

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 has a certain number of trials. Whether this worker is **successful** depends on the **average performance** over these trials.

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

Format:

taskId,description;uncertainty$difficulty%priorityLabel&workers:a list of worker IDs

Example:

123,image labelling;5$10%1&workers:0,1

Note: the 'priorityLabel' shows whether this task has a high priority or not. (1 denotes high priority and 0 denotes low priority)

2. Workers.txt

Format:

workerId,name%variability$ability;experienceLabel

Example:

0,Michael%-2$50;1

Note: the 'experienceLabel' shows whether this worker is senior or not. (1 denotes a senior worker and 0 denotes an ordinary worker)

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 a certain number of independent draws from this distribution plus a conditional value of 0 or 6. The number of draws is 5 if the task has a low priority and is 10 if it is a high-priority task. The conditional value is 0 if the worker is an ordinary worker and is 6 if the worker is senior.

To summarize, the average performance of a worker is:

The average of {5 or 10} independent performance scores drawn from a normal distribution, plus {0 or 6}. The values in the brackets {} depend on the type of the task and the worker.

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 worker information and 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) into the "**Output.txt**" file. Below are some sample outputs:

```
WorkerID    Name        Variability Ability     ExperienceLabel
0           Peter       -3          53          1
1           Sam         -2          82          0
2           Tommy       -7          61          0
3           James       -4          72          0
4           Amanda      -5          62          0
5           Hellen      -8          46          0
6           Kim         -3          71          0
7           Young       -4          62          1
8           Linda       -5          44          1
9           Ryan        -2          82          0
10          Jhon        -7          72          0

============================================================
processing taskId:  1
description :   image processing
uncertainty :   6
difficulty  :   10
priority    :   1
workers     :   7,0,8

    ------------------------------------------------------------
    Trial (1/3), Senior worker  : 7
    ------------------------------------------------------------
    The average performance is 58
    Assignment of Task 1 to worker 7 succeeds

============================================================
processing taskId:  2
description :   natural language processing
uncertainty :   7
difficulty  :   15
priority    :   1
workers     :   8,1,3

    ------------------------------------------------------------
    Trial (1/3), Senior worker  : 8
    ------------------------------------------------------------
    The average performance is 35
    Assignment of Task 2 to worker 8 fails

    ------------------------------------------------------------
    Trial (2/3), Ordinary worker  : 1
    ------------------------------------------------------------
    The average performance is 67
    Assignment of Task 2 to worker 1 succeeds

============================================================
processing taskId:  3
description :   computer vision
uncertainty :   5
difficulty  :   10
priority    :   0
workers     :   0,1

    ------------------------------------------------------------
    Trial (1/2), Senior workers : 0
    ------------------------------------------------------------
    The average performance is 49
    Assignment of Task 3 to worker 0 fails

    ------------------------------------------------------------
    Trial (2/2), Ordinary workers   : 1
    ------------------------------------------------------------
    The average performance is 72
    Assignment of Task 3 to worker 1 succeeds
```

Submission guidelines

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 be clear and concise, 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. Suppose your executable file is called `prog`, it should only take three additional arguments (Tasks.txt Workers.txt Output.txt) when you run the program:

$ ./prog Tasks.txt Workers.txt Output.txt

Important Notes (must read!):

1. Do not change any content in the provided files. The provided Tasks.txt and Workers.txt help test your codes. However, content in these files used for marking your code can be different.
2. Use structs to store information for tasks and workers.
3. 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 during marking will be not greater than 20.
4. Any standard input/output libraries (e.g., stringstream) can be used. You need to call some built-in libraries to handle delimiters from the file.
5. Your project need to have good commenting, have good layout, and have concise and self-explained codes.
6. Late submissions will be marked with a 25% deduction for each day, including days over the weekend.
7. Submissions more than four days late will not receive marks, unless an extension has been granted.
8. Academic misconduct is treated seriously. Any plagiarised work will be given a zero mark and reported to the University.