School of Computing & Information Technology

CSCI251/CSCI851 Advanced Programming Autumn 2020

Assignment 1 (Worth 12%)

Due 11:55pm Monday 20th April 2020. (Start of Week Six)

Overview

This assignment is to be implemented using procedural programming. The overall program should follow the processing of customer orders for robots from a robot construction company : **RAT** (Robots As Toys).

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 `capa.its.uow.edu.au` into the executable `RAT`. 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 capa 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 `RAT`, it must run as follows:

```
$ ./RAT Customers.txt Parts.txt Builders.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, except for `Parts.txt`, 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 customers, a list of parts, and a list of builders; all appropriately formatted so it's clear you have correctly partitioned the input data. It should be clear that you have correctly linked files, that should be clearer once you read the format of the data files. This report should go to standard out, not to `Output-file`. `Output-file` is used to report on results.

The customer orders in the customers file are to be processed in the order they are given.

The file `Output-file` should be ordered by customer and contain a clear report on the build plan, success or otherwise in the multiple attempts, as necessary.

When you start each customer you should report to standard out the customer name, the name of the order they are intending to be built, and the builder assigned. You should also report to standard out when a build succeeds or fails. The process below explains how multiple builds may be attempted, and those should be reported on.

The process involved in managing a customer order is as follows:

- For the given customer randomly allocate a builder who is constructing the robot.
- Determine the overall robot complexity as 20 plus the sum of the part complexities. For example:

  ```
  Sally:Snake:AEEEEEE.
  Overall robot complexity = 20+15 + 6 * 2 = 47
  ```

  If the complexity is greater than 100, it should be changed to exactly 100.
- Determine the overall robot variability as 5 plus the number of parts plus the variability of the builder. For example

  ```
  Sally:Snake:AEEEEEE.
  Reliable Rover:70:1.
  Overall robot variability = 5 + 7 + 1 = 13.
  ```

Report the customer name, project name, builder name, and the distribution parameters to the `Output-file`. Please report on the overall robot complexity too, that's your target.

- The builder attempts to build the robot. Generate a random value from a normal distribution with mean equal to the builder's ability and standard deviation equal to the overall robot variability just calculated.

  If the random value is greater than or equal to the overall robot complexity, the build is successful.

  If the random value is less than the overall robot complexity, the build fails.

- If the build succeeds you can move on to the next customer.
- If the build fails, the builder can re-attempt the build twice. For each re-attempt generate a random value from the same distribution as before, but for the first re-attempt add 5 and for the second re-attempt add 10.
- After three fails in total you should move to the next customer regardless of the result.
- The random value and success status for each attempt 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 the Moodle site.

The three data files are as follows, with the colons used to separate fields and the full stop to end the line.

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

   ```
   Customer name:Project name:List of parts.
   ```

   Example:

   ```
   Carly:Cat:ABCCCCE.
   Dodgy Dan:Dog:BCACECC.
   Ernie:Ettin:AABCCDD.
   Sally:Snake:AEEEEEE.
   ```

   The Customer name is the name of the entity that submitted the order. It is a non-empty string of printable characters that may include spaces.

   The Project name is chosen by the customer and is supposed to represent the type of robot to be built. Like the Customer name it is a non-empty string of printable characters that may include spaces.

   The List of parts contains a list of letters with each letter corresponding to a part in the parts file. They do not need to be in alphabetical order. There shouldn't be more than 10 parts for any order.

2. `Parts.txt`: This file corresponds to a publicly available catalogue for `RAT` so the content will be as provided here. You do not need to check the input validity but should check if the file is present.

   ```
   Part code:Part name:Minimum:Maximum:Complexity.
   ```

   ```
   A:Head:1:2:15.
   B:Torso:0:6:5.
   C:Leg:0:4:6.
   D:Arm:0:4:8.
   E:Tail:0:6:2.
   ```

   The part code will be a single capital letter. The minimum and maximum are limits on the number of that part type that is in proposed build. The complexity is used to determine how difficult a build will be depending on the parts added.

3. `Builders.txt`: No more than 5 entries.

   ```
   Name:Ability:Variability.
   ```

   Example:

   ```
   Reliable Rover:70:1.
   Sloppy Simon:20:4.
   Technical Tom:90:3.
   ```

   The name cannot be empty. The ability is an integer in the range 1 to 99 inclusive. The variability is an integer in the range 1 to 10 inclusive.

Output

If there is a problem with any 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 `capa.its.uow.edu.au` 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 to Moodle in a zip file `A1.zip`. There shouldn't be any directories or subdirectories within that zip file and you shouldn't submit an executable in it either.

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

1. The deadline is 11:55pm Monday 20^{th} April 2020. (Start of Week Six).
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.