CCPROG1 Term 1, AY 2021 – 2022

Machine Project Specifications

Groupings: Individual
Deadline: January 24, 2022 (Monday) 7:30 AM
Percentage: 20%

Deliverables:

`Zip` file containing:

- Program source codes – `c` and `h` files
- Test Script Documentation – `PDF` file

Submission guidelines: Submit the `zip` file to AnimoSpace
Filename format: `CCPROG1-MP-<Section>-<Surname>.zip`

SPECIFICATIONS

After being handed the Iron Throne, King Bran of House Stark, decided to devise a strategic plan to improve the economic growth of the Six Kingdoms. The Royal Treasury currently has 2,000 Golden Dragons (GD) only. To accomplish this task, he instructed his Hand, Lord Tyrion Lannister, and his Master of Coin, Lord Bronn of the Blackwater, to go to trade with other kingdoms in Westeros and other free cities in Essos. Various trading partners are shown in the sample opening screen of the game as displayed in Figure 1.

```
--- TRADING PARTNERS ---

      [1] Winterfell
      [2] Lys
      [3] Myr
      [4] Pentos
      [5] Qohor
      [6] Volantis

      [9] Quit

Choice: _
```

Figure 1. Opening screen

Each trading partner in The Known World sells a common set of wares and goods. These items can be traded as shown in the sample screen design below in Figure 2.

```
------------------------------------------------------------------
                            WINTERFELL
------------------------------------------------------------------
ITEM        WARES AND GOODS               SELLING/BUYING PRICE
------------------------------------------------------------------
[1]         Sweet Beet                          190
[2]         Timber                              220
[3]         Intricate Lace                      385
[4]         Intoxicating Perfume                489
[5]         Pale Amber Wine                     599
[6]         Myrish Eye                          695
[7]         Qohorik Tapestry                    780
[8]         Valyrian Steel                      898
------------------------------------------------------------------

Days Remaining:         15 Days
GD:                     2000
Debt:                   0
Savings:                0
Capacity:               0 / 100

[B]uy       [S]ell      [W]heelhouse      [I]ron Bank       [Q]uit

Choice: _
```

Figure 2. Sample screen for trading

Aside from the price of each item, the trading screen should display the following:

- Days Remaining
- Golden Dragons (GD)
- Debt
- Savings
- Capacity

It should also allow the user to access other options such as (1) riding the wheelhouse, (2) going to the Iron Bank of Braavos, and (3) quitting the game.

The buying and selling prices of wares and goods in each trading partner varies depending on the scale shown in Table 1. Each trading partner has a special item which they trade for a special price, as listed in Table 2.

Wares and Goods	Normal Price	Special Price
Sweet Beet	180 – 200	100 – 150
Timber	280 – 300	200 – 250
Intricate Lace	380 – 400	300 – 350
Intoxicating Perfume	480 – 500	400 – 450
Pale Amber Wine	580 – 600	500 – 550
Myrish Eye	680 – 700	600 – 650
Qohorik Tapestry	780 – 800	700 – 750
Valyrian Steel	880 – 900	800 – 850

Table 1. Range of prices per wares and goods

Trading Partner	Special Item
Winterfell	Timber
Lys	Intoxicating Perfume
Myr	Myrish Eye Intricate Lace
Pentos	Pale Amber Wine
Qohor	Qohorik Tapestry Valyrian Steel
Volantis	Sweet Beet

Table 2. List of specialty items for each trading partner

Limitations

Lord Hand Tyrion and Lord Master Bronn have only 15 days to go around and accumulate wealth for The Royal Treasury. Tyrion and Bronn can travel from place to place using the wheelhouse. Every time they travel using the wheelhouse, they lose 1 day. To make the game user-friendly, the game should allow the player to exit whenever he wants.

The wheelhouse also has enough storage to store up to 50 wares and goods. When they visit another trading partner, they may randomly meet a merchant to increase the capacity of the wheelhouse by 50 which costs 200 GD. The merchant appears in any trading partner 10% of the time.

The Iron Bank of Braavos

The Iron Bank of Braavos is the most powerful bank in The Known World. King Bran instructed Lord Hand Tyrion and Lord Master Bronn to transact with the Iron Bank of Braavos whenever needed. They may deposit GDs to gain interest. They may also get a loan to increase their trading power. The bank has a branch in every trading partner; thus, they can drop by anytime without missing a day.

In the bank, they can do the following:

- Deposit GDs (indicate the amount of GDs to be deposited)
- Withdraw GDs (indicate the amount of GDs to be withdrawn)
- Get a loan (indicate the amount of GDs to be loaned)
- Pay-back a dept (indicate the amount of GDs to pay-back)

The bank gives daily interest of 10% to its customers. At the start, the account of The Royal Treasury has 0 GDs. Additionally, loans also have interests. For The Iron Bank of Braavos, the daily interest is 5%.

Ending the Game

When the user exits the game or when the 15 days are over, the game should display the following:

- Days Remaining
- Golden Dragons (GD)
- Debt
- Savings
- Capacity
- Inventory for each ware and goods

Solving the Machine Project

Step 1. Problem analysis and algorithm formulation

Read the MP Specifications again. Clearly identify the required information from the user, the needed processes, and the output(s) of your program. Clarify with your professor any issues that you might have regarding the machine project.

When you have read all the necessary information, identify the necessary functions that you need to modularize the project. Identify the parameters and the return values of these functions. Write your algorithm for each of these modules/functions as well as the algorithm for your main program.

Step 2. Implementation

In this step, you are to translate your algorithm into proper C statements. While implementing, you are to perform the other phases of program planning and design (discussed in the other steps below) together with this step. You may choose to type your program in a text editor or an IDE (i.e., Dev-C IDE) at this point. Note that you are expected to use statements taught in class. You can explore other libraries and functions in C if you can clearly explain how these works. You may also use arrays if you can justify your implementation. For topics not covered, it is left to the student to read ahead, research, and explore by himself.

Note though that you are NOT ALLOWED to use:

- Global variables
- Static variables
- `goto` statements
- `break`, `exit()`, or `return` statement to prematurely terminate a loop or to terminate the function program.
- Calling the `main()` function to repeat the process instead of using loops.

It is best that you perform your coding "incrementally."

- Dividing the program specification into subproblems and solving each problem separately according to your algorithms.
- Code the solutions to the subproblems one at a time. Once you are done coding the solution for one subproblem, apply testing and debugging.

Step 3. Documentation

While coding, you must include internal documentation in your program. You are REQUIRED to have the following:

- Introductory comments
- Function comments
- Inline comments

Introductory comments are found at the very beginning of your program before the preprocessor directives. Follow the format shown below. Note that items in between <> should be replaced with the proper information.

```
/*
     Description:      <Describe what this program does briefly>
     Programmed by:    <your name here>   <section>
     Last modified:    <date when last revision was made>
    [Acknowledgements: <list of sites or borrowed libraries and sources>]
*/

<Preprocessor directives>

<function implementation>

int main() {
   return 0;
}
```

Figure 3. Format for introductory comment

Function comments precede the function header. These are used to describe the function, each of its parameters, and the return value. If applicable, include pre-conditions as well. Pre-conditions refer to the assumed state of the parameters. Follow the format below when writing function comments:

```
/*   <Description of function>
     Precondition: <precondition / assumption>
     @param    <name>   <purpose>
     @return   <description of returned result>
*/
<return type> <function name> (<parameter list>) {
...
}
```

Figure 4. Format for function comments

Example of function comment:

```
/*    This function generates the buying or selling price of a specific item
      between the bounded values

      Precondition: nLow and nUpper should be positive values

      @param  nLow is the lower bound of the price
      @param  nUpper is the upper bound of the price

      @return random price that was generated
*/
int getRandomPrice (int nLow, nUpper) {
...
}
```

Figure 5. Example of a function comment

Inline comments are other comments in major parts of the code. These are expected to explain the purpose or algorithm of groups of related code, especially for long functions.

Step 4. Testing and Debugging

For each feature of your program, fully test it before moving to the next feature. Below are sample questions for testing your code:

- What should be displayed on the screen if the user provided an input?
- What would happen if the user provided a borderline value (minimum and maximum values of a given range)?
- What would happen if the user provided incorrect inputs (e.g., values not within the range)?
- Is my program displaying the correct output?
- Is my program following the correct sequence of events?
- Is my program terminating correctly? Does it exit when the user enters the command to quit? Does it exit when the program's goal has been met? Is there an infinite loop?

TEST SCRIPT DOCUMENT

Open the file **`Documentation.docx`**. Edit this file for the test script documentation of your project. Test script should be in a table format, with header as shown in Table 3 below. There should be at least 3 distinct test classes (as indicated in the description) per function. There is no need to test functions which are only for screen design.

Function Name: `getTotalPrice()`					
Test #	**Test Description**	**Sample Input**	**Expected Result**	**Actual Result**	**P/F**
1	fAmt and nQty are positive	fAmt = 30.00 nQty = 3	…	…	…
2	…	…	…	…	…
3	…	…	…	…	…

Table 3. Sample table in test script document

Using the predefined pseudorandom function of C

Refer to the following sample program

```
#include <stdio.h>
#include <stdlib.h> /* where srand and random are defined */
#include <time.h> /* where time function is defined */

int main() {
      int nValue, nBounded;
      int nUpperbound = 3000, nLowerBound = 1000;
      int i;

      /* initialize a random sequence based on time */
      srand(time(NULL));

      /* generate a pseudorandom number from 0-122; rand returns an int */
      nValue = rand() % 123;
      printf("Random: %d\n", nValue);

      /* generates a number between a lower bound and an upperbound */
      nBounded = rand() % (nUpperbound – nLowerbound + 1) + nLowerbound;
      printf("Random value within a range: %d", nBounded);
}
```

Figure 6. A C program using the pseudorandom function

IMPORTANT POINTS TO REMEMBER:

- You are required to implement the project using the C language (NOT C++). Make sure you know how to compile and run in both the IDE (DEV-C++) and the command prompt (via `gcc <yourMP.c> -o <yourExe.exe>`).
- The implementation will require you to:
    - Create and use functions.
    - Appropriately use conditional statements, loops, and other constructs discussed in class. Do not use brute force solutions.
    - Consistently employ coding conventions.
    - Include internal documentation.
- Non-use of self-defined functions will result in a grade of 0 for the machine project.
- After the deadline of the project, the submission facility will be locked and thus no MP will be accepted anymore. Non-submission will result in a grade of 0 for the MP.
- You will demonstrate your project on a specified schedule during the term. Being unable to show up on time during the demo or being unable to answer convincingly the questions during the demo will result in a grade of 0 for the MP. The project is initially evaluated via black box testing (i.e., based on the output

of the running program). Thus, if the program does not compile successfully using `gcc` and execute in the command prompt, a grade of 0 for the project will be incurred. However, a fully working project does not ensure a perfect grade, as the implementation, including correctness and code compliance, is still checked.

- Any requirement not fully implemented, and instruction not followed will result in deductions.
- A maximum of 10 points may be given for features over & above the requirements, such as use of arrays for storing item counts, impressive user interface (i.e., using arrow keys), list of high scorers, & other features not conflicting with the given requirements, or changing the requirements, subject to evaluation of the professor. Required features must be completed first before bonus features are credited. Note that use of `conio.h`, string variables, or other advanced C commands/statements will not necessarily merit bonuses
- This is an individual project. Working in collaboration, asking for other people's help, and/or copying other people's work are considered cheating. Cheating is punishable by a grade of 0.0 for the course, aside from which, a cheating case may be filed with the Student Discipline Formation Office (SDFO).

DELIVERABLES

Submit a zip file containing the source code files and a PDF file of the documentation via AnimoSpace. **DO NOT INCLUDE ANY EXECUTABLE FILE in your zip file submission.**

HONESTY POLICY AND INTELLECTUAL PROPERTY RIGHTS

Honesty policy applies. Please take note that you are **NOT allowed to borrow and/or copy-and-paste** – in full or in part any existing related program code from the internet or other sources (such as printed materials like books, or source codes by other people that are not online). You should develop your own codes from scratch by yourself.

According to the handbook (5.2.4.2), "faculty members have the right to demand the presentation of a student's ID, to give a grade of 0.0, and to deny admission to class of any student caught cheating under Sec. 5.3.1.1 to Sec. 5.3.1.1.6. The student should immediately be informed of his/her grade and barred from further attending his/her classes.