CCPROG1 MACHINE PROJECT

Term 1, AY 2020 – 2021

due on Jan. 25, 2021 (2359)

Oil-bularyo ON THE GO is a turn-based strategy game in which the player assumes the role of a travelling salesman. The travelling salesman regularly goes on a sales trip consisting of 15 days. The objective of the game is for a player to make the most money on one sales trip/cycle.

A players earns or losses money from the sale/purchase of raw materials (the essential oils), namely, lemon, rosemary, mint and lavender and/or the sale of diffuser oils.

The premise of this game is that the player is living in Manila. The player travels to other cities like Makati, Alabang, and Quezon City. Traveling to another city takes one day.

The player begins with $1500 cash on hand, and a $4000 loan from the loan shark. The loan shark charges a daily interest to unpaid loans.

In each city (or everyday), the player checks the prices of the different essential oils and decides to buy or sell oils. The player can also make and sell diffuser oils. Recipe for the diffuser oils are listed below.

Diffuser Oils

Clear Minds	Energy Booster	Calming
3 drops Lemon	1 drop Rosemary	3 drops Lavender
2 drops Rosemary	1 drop Mint	2 drops Rosemary
2 drops Mint	1 drop Lemon	1 drop Lemon

Diffuser oils are made using the processing plants of each city. The use of the processing plant is subject to a fee that varies depending on the city. The fees charged by each city are as follows:

Cities and Charge for making Diffuser Oils in that City	
Manila	20 – 30
Makati	80 – 100
Alabang	70 – 90
Quezon City	40 – 60

Prices and stocks of essential oils for each city varies. The selling price of Diffuser oils is computed based on the total price of the raw materials plus the Diffuser Oil Premium Rate (i.e. 15%) of that total.

The player can choose to pay-off some or all of his debt.

There are days when the player is lucky and gets free essential oils or diffuser oils.

Your task: Implement this game in C. The game is menu-based. This means that options of what the user can do will be listed, and the user will choose which action to take.

At the start of the game, a player can choose one of the following options:

- Start a Sales Trip
- View the Top Ten List
- Exit the Game

At the end of each sales trip/cycle (a sales trip ends after 15 turns), a summary of the remaining essential oils, diffuser oils, cash and debt is displayed. The game maintains a top 10 list. If the total cash on hand after is higher than the top 10, the player will be asked to enter his/her name and the name and the score will be added to the top 10 list.

At the end of each sales trip/cycle the player will be presented with a menu with the 3 options above. The game will end only after the user choses to "Exit the Game".

When the player is in a city, the following must be shown:

- Day #
- Cash
- Debt
- For each essential oil that is available in that city, show
  - the price per bottle
  - the price per drop
  - if the player has stock-on-hand, show number of drops @average cost per drop
- For each diffuser oil, show
  - price per bottle
  - if the player has stock-on-hand, show number of bottles @average price per bottle

The player can

- choose an action: **Buy**, **Sell**, **Make**, **Pay Debt**, **Loan**, or **Travel**
  - for **Buy** and **Sell** actions,
    - the player can choose which essential oil or diffuser oil
    - the maximum possible bottles that the user can buy or sell should be shown

        - note that the player must enter a valid value. The player can enter a 0 to not execute this transaction.
    - for **Make** action,
        - the player can choose which diffuser oil to make
        - only the diffuser oil that can be made, based on the player's current inventory should be shown.
        - the maximum possible bottles that the user can make should be shown
        - note that the player must enter a valid value. The player can enter a 0 to not make a diffuser oil.
    - for **Pay Debt** and **Loan** actions,
        - the player will indicate how much to pay or loan
        - note that if the value entered is not a valid amount, nothing happens
    - for **Travel** action,
        - the cities where the player can go to are shown
        - the player must provide a valid destination

Prices

- All amounts are whole number amounts only. The fractional part of the amount is **dropped**.
- Amounts of the different products are generated randomly each day.

Essential Oils	Price Ranges per bottle (10 drops / bottle)
Lemon	100 – 150
Lavender	20 – 60
Rosemary	70 – 100
Mint	130 – 200

To generate the price for each Essential Oil,

1. randomly generate a value based on its price range
2. randomly determine if the charge will be added, subtracted, or be multiplied to this value

Operator	Chance
add	35%
subtract	35%
multiply by a factor	20%
retain the generated value	10%

The factor is a randomly generated value between 101 and 200.

The selling price for each Diffuser Oil is computed based on the current prices of the essential oils as:

$$1.15 \times \sum_{i \in ingredients} (drop_i \times pricePerDrop_i)$$

Other Game Settings

At the start of a new game	
Cash	1500
Loan	4000
Current Location	Manila

Game Settings	
Days	15
Loan Shark Rate	10%
Diffuser Oil Premium	15%
Freebie chance	EO (40%), Diffuser Oil (30%)

Cities and Charge for making Diffuser Oils in that City	
Manila	20 – 30
Makati	80 – 100
Alabang	70 – 90
Quezon City	40 – 60

For generating random numbers:

For random number generation, refer to https://www.geeksforgeeks.org/rand-and-srand-in-ccpp/ on how to use rand() and srand().

Screens:

1. It is up to the programmer on how each screen will look like.
   - The screens MUST be neat and organized.
   - All the necessary details are displayed, and easy to find.
2. This is a menu-based application. DO NOT include Yes or No questions, e.g. Do you wish to continue?, Are you sure?, Play another game?, Do you want to visit a city?, etc.

On documentations and coding standards:

1. Follow coding standards.
2. Include internal documentations.
3. Include function specifications in the source code. Before each of your functions, include the comments containing function specifications. Follow the format below.

```
/*
  Description:   what the function does
  Parameters:
       param1    what this param1 is for
       param2    description of param2
          :
  Return value:  description
 */
```

4. Create a Test Case document, save as PDF. For each function, identify appropriate test cases/scenarios. Test and validate your functions.

   Function: *function declaration/signature*

Test Description	Input value/parameters	Expected output/result	Actual output/result	Pass/Fail
Case 1 desc				P or F
Case 2 desc				
:				

Important Notes:

1. All source codes (with internal documentation and function specs) and Test Case documents (pdf file) MUST be uploaded in AnimoSpace before **January 25, 2021 (2359)**.
2. For all submissions made, the following are assumed:
   a. You worked on the entire project by yourself;
   b. the submitted files contain your original work; and
   c. you did not share a part or the entire source code to other people.
3. This is an **individual** project. Any form of cheating (working in collaboration, asking other person's help to accomplish a part or the entire MP, copying any code from another person's work or post) will merit a grade of 0.0 for the CCPROG1 and a discipline case.
4. You will present your project during a scheduled demo. The schedule will be announced later. During the demo, the submitted code will be downloaded, compiled, and executed.
5. Your solution must include use of functions, conditional, and iterative statements.
6. You will incur deductions when:
   a. The return statement used in a void function.
   b. Minimal use of user-defined functions.
   c. Features are either missing or not fully implemented, or instructions are not followed.
7. The following will automatically make your MP score 0:
   a. Not submitting before the deadline. AnimoSpace flagged your submission as LATE.
   b. Use of global variables.
   c. Use of keywords such as `goto`, `continue`, and statements such as `System.exit()`.
   d. Calling the `main()` function to make the program execute again.
   e. Use of `break` statements in a non-switch block.
   f. No user-defined functions are used. Your program is just one big `main()` function.
   g. Compilation error encountered during the demo.
   h. No visible outputs on the screen.