Food Magnate Simulation

Programming Tasks

The following require you to open the skeleton program and make modifications

Task 1 (max. 6 marks)

This question refers to the subroutine `ModifyCompany` within the `Simulation` class.

Currently, the user is prompted to enter a value of 1, 2 or 3, but if nothing is entered by the user, the program responds by outputting a blank line.

Change the subroutine `ModifyCompany` to present the user with an additional choice: "C. Cancel". If the user enters anything other than 1, 2, 3 or an upper-case 'C', the menu should be redisplayed repeatedly until either 1, 2, 3 or C is selected. If 1, 2 or 3 is entered, `ModifyProgram` should behave as normal. If an upper-case 'C' is entered, the program should output 'Operation Cancelled', and `ModifyCompany` should terminate without executing any additional code.

Test that the changes you have made work:

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- enter D at the next prompt for default companies
- enter 3 for 'modify company'
- enter 'AQA Burgers' when prompted for a company name
- enter 'X' at the first prompt of the 'modify company' submenu
- enter 'C' at the second prompt of the 'modify company' submenu

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `ModifyCompany`

b. SCREEN CAPTURE(S) showing the required test

Task 2 (max. 9 marks)

This question refers to the subroutine `GetRandomLocation` within the `Settlement` class.

This subroutine generates a random location within the bounds of the settlement that is used to position a new household. Currently, there is no mechanism for ensuring that a new household is not assigned the location of an existing household.

Change the subroutine `GetRandomLocation` to ensure that only unoccupied locations are returned. Prior to returning the location, a check should be made to determine whether the location is already occupied by a household. If it is already occupied, a new location should be generated, repeatedly if necessary.

Test that the changes you have made work:

- modify the Settlement constructor in the following ways:
  - change `XSize = 1000` to `XSize = 3`
  - change `YSize = 1000` to `YSize = 3`
  - change `StartNoOfHouseholds = 250` to `StartNoOfHouseholds = 8`

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- enter D at the next prompt for default companies
- enter 1 for 'display details of households'

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `GetRandomLocation`

b. SCREEN CAPTURE(S) showing the required test

Task 3 (max. 7 marks)

This question refers to the subroutine `ExpandOutlet` within the `Company` class, as well as a new subroutine, `ExtendCapacity`, within the `Outlet` class.

Currently, each outlet has a limit, beyond which it cannot be expanded. Any attempt to expand beyond the limit results in the capacity being set at the limit itself.

Create a new subroutine in the `Outlet` class called `ExtendCapacity` which takes a single integer parameter. When this subroutine is called, the value of the attribute `MaxCapacity` should be multiplied by the value of this parameter.

Change the `ExpandOutlet` subroutine so that, in the event of an attempt to increase an outlet beyond its `maxCapacity` value, a random number is generated. Using this number, there should be a 40% chance of calling the new subroutine, `ExtendCapacity`, and passing a value of 2, a 35% chance of calling it and passing a value of 3, and a 25% chance of calling it and passing a value of 4. The message 'Outlet max capacity expanded' should be displayed in these circumstances, with no other message.

Test that the changes you have made work:

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- enter D at the next prompt for default companies
- enter 2 for 'display details of companies', ensuring that outlet 4 for 'Paltry Poultry' is visible in your screenshot
- enter 3 for 'modify company'
- enter 'Paltry Poultry' when prompted for a company name
- enter 3 for 'expand outlet'
- enter 4 when prompted for an ID
- enter 1000 when asked for an amount
- enter 2 for 'display details of companies', ensuring that outlet 4 for 'Paltry Poultry' is visible in your screenshot

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `ExpandOutlet`

b. Your PROGRAM SOURCE CODE for the new subroutine `ExtendCapacity`

c. SCREEN CAPTURE(S) showing the required test

Task 4 (max. 8 marks)

This question refers to the subroutine `ProcessDayEnd` within the `Simulation` class, as well as a new subroutine, `ProcessLeavers`, within the `Settlement` class.

Currently, there is no mechanism for households to leave a settlement.

Create a new subroutine in the `Settlement` class called `ProcessLeavers`. This subroutine should accept no parameters and should return an integer value. Each household in the settlement should be subject to a random 2% chance of leaving the settlement. The `ProcessLeavers` subroutine should remove households as applicable from the `Households` data structure, and return the number of households that were removed.

Modify the subroutine `ProcessDayEnd`, so that the final instructions are to call `ProcessLeavers` and output the number of households that left the settlement.

Test that the changes you have made work:

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- enter D at the next prompt for default companies
- enter 6 for 'advance to next day'

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `ProcessDayEnd`

b. Your PROGRAM SOURCE CODE for the new subroutine `ProcessLeavers`

c. SCREEN CAPTURE(S) showing the required test

Task 5 (max. 7 marks)

This question refers to the subroutine `Run` within the `Simulation` class.

Currently, when modifying a company, the user needs to enter a company name in full, for which they must either remember it or scroll up the console window to see it previously displayed.

Change the subroutine `Run` so that when the user enters option 3 from the main menu (modify company), they are presented with a numbered list of names of companies. The first company to be displayed in the list should be displayed next to a number 1, even though its index in the `Companies` data structure will be 0.

When the user enters the number next to the company name, the program should respond in the same way as it would have done had the company's name been entered.

Entering the company's name should no longer be effective, and the input message should read 'enter the number next to the company you wish to modify'.

Test that the changes you have made work:

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- enter D at the next prompt for default companies
- enter 3 for 'modify company'
- enter 3 when asked for a number
- enter 3 for 'expand outlet'
- enter 4 for the ID
- enter 1000 for the amount
- enter 2 for 'display details of companies'

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `Run`

b. SCREEN CAPTURE(S) showing the required test

Task 6 (max. 5 marks)

This question refers to an additional constructor for the `Outlet` class.

Currently, when a new outlet is constructed, this takes place by way of X and Y coordinates being passed to the `Outlet` constructor. In this task, you will create an additional constructor that facilitates random placement.

Without making any changes to the existing constructor, create an additional constructor for the `Outlet` class. This constructor should take as parameters three integers – `MaxX`, `MaxY` and `MaxCapacityBase` – and a Boolean, `IsRandom`. The `MaxX` and `MaxY` integers represent the highest values for X and Y that an outlet could possibly have within its settlement.

The constructor should contain code that sets `XCoord` to a random integer value between 0 and `MaxX` inclusive, as well as code that sets `YCoord` to a random integer value between 0 and `MaxY` inclusive. All other attributes of the `Outlet` class should be set identically to those of the existing constructor.

Test that the changes you have made work:

- Modify the call to the `Outlet` class's constructor within the subroutine `OpenOutlet` of the `Company` class; it should read as follows:
  - `Dim NewOutlet As New Outlet(5, 10, Capacity, True)`
- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- enter D at the next prompt for default companies
- enter 2 for 'display details of companies'

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the new `Outlet` constructor

b. SCREEN CAPTURE(S) showing the required test

Task 7 (max. 7 marks)

This question refers to the subroutines `DisplayMenu` and `Run` within the `Simulation` class.

Currently, the program allows the user to advance the simulation for multiple days only by repeatedly selecting option 6 from the menu.

Change `DisplayMenu` to include an additional option: '5. Advance'.

Change `Run` so that if option 5 is selected, the user is prompted for the number of days they wish the simulation to advance. This number, which does not require validation, will be the number of times that the subroutine `ProcessDayEnd` is called.

Test that the changes you have made work:

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- enter D at the next prompt for default companies
- enter 5 for 'advance'
- enter 3 when asked for a number

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `Run`

b. Your PROGRAM SOURCE CODE for the amended subroutine `DisplayMenu`

c. SCREEN CAPTURE(S) showing the required test

Task 8 (max. 9 marks)

This question refers to the subroutine `AddCompany` within the `Simulation` class.

Functionality will be added to allow a company type to be assigned at random.

Change the `AddCompany` subroutine so that the user is prompted to enter 1, 2, 3 or 4. The existing prompt should be updated to indicate that '4' means a randomly chosen type of company.

If 1, 2 or 3 is entered, the program should continue as before. If '4' is entered, a random number should be used to determine which of the three types of company will be created. Each type of company should be equally likely to be created.

Test that the changes you have made work:

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- leave the second prompt blank, to indicate user-defined companies
- enter 1 when asked for a number of companies
- enter the company name 'Random Restaurant'
- enter a starting balance of 50000
- enter 4 to indicate a 'random' new company
- enter 2 for 'display details of companies'

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `AddCompany`

b. SCREEN CAPTURE(S) showing the required test

Task 9 (max. 6 marks)

This question refers to the subroutine `ProcessDayEnd` in the `Company` class, as well as a new subroutine, `CloseAllOutlets`, also in the `Company` class.

Currently, a company can continue operating irrespective of how far below zero their balance falls.

Create a new subroutine called `CloseAllOutlets`, in the `Company` class, which iterates through all outlets belonging to the company, and closing them with a call to `CloseOutlet`.

Modify the subroutine `ProcessDayEnd` in the `Company` class, so that immediately before the return statement, the value of the `balance` field is checked. If it is below zero, a call to `CloseAllOutlets` is made, and a message is output to notify the user that all outlets have closed.

Test that the changes you have made work:

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- leave the second prompt blank, to indicate user-defined companies
- enter 1 when asked for a number of companies
- enter the company name 'Bankrupt Burgers'
- enter a starting balance of 1000
- enter 2 to indicate a family restaurant
- enter 6 in the main menu, 'advance to next day'
- enter 2 in the main menu, 'display details of companies'

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `ModifyCompany`

b. Your PROGRAM SOURCE CODE for the new subroutine `CloseAllOutlets`

c. SCREEN CAPTURE(S) showing the required test

Task 10 (max. 11 marks)

This question refers to a new class, called `FoodTruck`, as well as the subroutine `ProcessDayEnd` in the `Company` class.

Create a new class, called `FoodTruck`, which inherits from the class `Outlet`. As well as the inherited subroutines and attributes, `FoodTruck` should include a new subroutine called `Move`. Movement should take place in a random direction, moving one 'square' north, south, east or west. There is no need to validate the food truck's movement, which is permitted to leave the settlement as a result of its `XCoord` and `YCoord` values being beyond the settlement's bounds.

The constructor for `FoodTruck` should take `XCoord` and `YCoord` integers as parameters, then pass these to the superclass constructor along with a value of 10 for capacity.

Change `ProcessDayEnd` so that the `move` subroutine is called for any outlets that are of type `FoodTruck`.

Test that the changes you have made work:

- Modify the call to the `Outlet` class's constructor within the `OpenOutlet` subroutine of the `Company` class; it should read as follows:

  ```
  Dim NewOutlet As New FoodTruck(X, Y)
  ```

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- leave the second prompt blank, to indicate user-defined companies
- enter 1 when asked for a number of companies
- enter the company name 'Taco Truck'
- enter a starting balance of 30000
- enter 1 to indicate a fast-food restaurant
- enter 2 to display details of companies
- enter 6 to advance to the next day
- enter 2 to display details of companies

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the new class `FoodTruck`

b. Your PROGRAM SOURCE CODE for the amended subroutine `ProcessDayEnd`

c. SCREEN CAPTURE(S) showing the required test, ensuring that the location of the food truck is visible after each entry of '2' in the main menu

Task 11 (max. 15 marks)

This question refers to the subroutine `GetIndexOfCompany` within the `Simulation` class.

Currently, when a company is searched for using this subroutine, the whole company name is required in order to generate a match.

Change `GetIndexOfCompany` so that if the user enters a search term that is contained within the name of one company, the index of that company is returned. If the text is contained within the names of multiple companies, the user should be presented with all matching company names before being asked to type one of them in full in order to select it.

The subroutine should continue to be non-case-sensitive, and a search for a company that finds nothing, or an attempt to select a matching company that doesn't actually match one of the search results, should still return a value of -1.

Test that the changes you have made work:

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- enter D at the next prompt for default companies
- enter 3 for 'modify company'
- enter a lower-case 't' for the company name
- type 'Paltry Poultry' when asked to type the name of a company

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `GetIndexOfCompany`

b. SCREEN CAPTURE(S) showing the required test

Task 12 (max. 7 marks)

This question refers to the subroutine `CloseOutlet` in the `Company` class.

Currently, closing an outlet incurs no expense on the part of the company.

Change `CloseOutlet` so that a company's balance decreases for each outlet that is closed. The cost of closing the outlet depends on both the type of the company and the capacity of the outlet being closed. Taking 'capacity' as being the number of seats in an outlet, the costs of closing an outlet are as follows:

Fast-food outlet: 75 per seat

Family outlet: 50 per seat

Named chef outlet: 150 per seat

Test that the changes you have made work:

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- enter D at the next prompt for default companies
- enter 2 for 'display details of companies
- enter 3 for 'modify company'
- enter 'Paltry Poultry'
- enter 2 for 'close outlet'
- enter 4 when prompted for an ID
- enter 2 for 'display details of companies'

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `CloseOutlet`

b. SCREEN CAPTURE(S) showing the required test, ensuring all details for Paltry Poultry's outlets are visible after each request of 'display details of companies'

Task 13 (max. 8 marks)

This question refers to the subroutines `DisplayMenu` and `Run` in the `Simulation` class.

Currently, it is possible to add a company to a simulation, but not to remove one.

Change `DisplayMenu` to include an additional option: '5. Remove company'.

Change `Run` so that if option 5 is selected, the user is prompted for the name of the company they wish to remove. After the user has entered the name of the company, that company's index should be obtained via `GetIndexOfCompany`. The program should repeatedly ask them for a company name until either a valid name has been entered, or 'cancel' (any combination of upper case and lower case) has been entered.

If 'cancel' is entered, the user should be returned to the main menu. Otherwise, the company with a name matching the user entry should be removed from the simulation, and the user should be returned to the main menu.

Test that the changes you have made work:

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- enter D at the next prompt for default companies
- enter 5 for 'remove company'
- type CANCEL (all in upper case) when prompted for the name of a company
- enter 5 for 'remove company'
- type 'Paltry Poultry' when prompted again for the name of a company
- type 2 for 'display details of companies'

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `DisplayMenu`

b. Your PROGRAM SOURCE CODE for the amended subroutine `Run`

c. SCREEN CAPTURE(S) showing the required test

Task 14 (max. 20 marks)

This question refers to the subroutines `DisplayMenu` and `Run` in the `Simulation` class, as well as two new subroutines: `RunToTarget` in the `Simulation` class, and `GetBalance` in the `Company` class.

Currently, the simulation runs day by day, regardless of the effects of any changes.

Create a new subroutine in the `Company` class called `GetBalance`, which should accept no parameters and return the value of the balance attribute.

Create a new subroutine in the `Simulation` class called `RunToTarget`. This subroutine should prompt the user for upper and lower limits, storing them in integer variables called `UpperLimit` and `LowerLimit`. No validation is required for user input.

The simulation should run, via repeated calls to `ProcessDayEnd` in the `Simulation` class, until one company has a balance either equal to or above `UpperLimit` or equal to or below `LowerLimit`, using calls to the new subroutine `GetBalance`. At this point, there should be no additional calls to `ProcessDayEnd`, and the program should output the name of the company, its balance and the number of days that have elapsed since the beginning of the simulation.

In the event that multiple companies reach `UpperLimit` and/or `LowerLimit` at the same time, the program need only display the details of one of the companies.

Change `DisplayMenu` to include an additional option: '5. Run to target'.

Change `Run` so that if the user enters option 5, a call is made to `RunToTarget`.

Test that the changes you have made work:

- run the **Skeleton Program**
- leave the first prompt blank, to indicate a normal-sized settlement
- enter D at the next prompt for default companies
- enter 5 for 'run to target'
- enter 0 when prompted for the lower limit
- enter 100000 (one hundred thousand) when prompted for the upper limit

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended subroutine `DisplayMenu`

b. Your PROGRAM SOURCE CODE for the amended subroutine `Run`

c. Your PROGRAM SOURCE CODE for the new subroutine `RunToTarget`

d. Your PROGRAM SOURCE CODE for the new subroutine `GetBalance`

e. SCREEN CAPTURE(S) showing the required test; only the final 'events' section needs to be included

Task 15 (max. 14 marks)

This question refers to a new class called `CitySettlement`, as well as the constructor of the `Simulation` class.

Currently, a new `Settlement` object is constructed by way of calls to the constructor of either `Settlement` or `LargeSettlement`.

Create a new class called `CitySettlement`, which inherits from `LargeSettlement`. Its constructor should have the same parameters as the `LargeSettlement` constructor, and should call the `LargeSettlement` constructor, passing on its own parameters.

`CitySettlement` should include a subroutine called `AddHousehold`, which should override the `AddHousehold` subroutine of the `Settlement` class. This new subroutine should select a random location within the city, then select a random integer between 2 and 20 inclusive. This number of new households should then be created and added to the settlement at the single location generated.

This is to model families living in apartment blocks. In this class, creating a settlement with 100 households should have the effect of creating 100 such apartment blocks, with each block containing between 2 and 20 households. This means that a 250-household `CitySettlement` will contain more than 250 households, as it should contain 250 blocks of households.

Change the constructor of the `Simulation` class to present the user with a choice of 'a large settlement', 'a normal-sized settlement' or 'a city settlement' (the order is unimportant). If a city settlement is selected, the user should be prompted for additional x-size, y-size and households values. These values should then be sent as parameters to a call to the new `CitySettlement` constructor.

Test that the changes you have made work:

- run the **Skeleton Program**
- at the prompt, indicate a city settlement
- enter a value of zero for each of the additional prompts
- enter D for default companies
- enter 1 for 'Display details of households'

Evidence that you need to provide:

a. Your PROGRAM SOURCE CODE for the amended `Simulation` constructor

b. Your PROGRAM SOURCE CODE for the new class `CitySettlement`

c. Your PROGRAM SOURCE CODE for the `AddHousehold` subroutine of the `Settlement` class

d. SCREEN CAPTURE(S) showing the required test, ensuring at least the last five households are visible