Food Magnate Simulation

MARKS **/55**

Programming Theory Questions

These questions refer to the preliminary material and require you to load the skeleton program, but <u>do not</u> require any additional programming.

1. State the name of an identifier for:

   a) An attribute in the `Settlement` class that is **only** accessible to subclasses of `Settlement` [1]

   .....

   b) A subroutine in the `Settlement` class that returns something other than a primitive value [1]

   .....

   c) A subclass [1]

   .....

   d) A local variable that is used to return a Boolean [1]

   .....

   e) A subroutine in the `Company` class that **cannot** be called from outside the `Company` class [1]

   .....

   f) A library string function called from the `GetIndexOfCompany` subroutine in the `Simulation` class [1]

   .....

   g) A collection attribute in the `Company` class [1]

   .....

   h) An instance of `Settlement` [1]

   .....

2. Showing and explaining your working, give the probability of a call to `ProcessCostOfFuelChangeEvent` being made from the `DisplayEventsAtDayEnd` subroutine in the `Simulation` class. [3]

   .....

   .....

   .....

   .....

   .....

3. Explain how validation might be added to the `OpenOutlet` subroutine of the `Company` class to prevent a new outlet being created beyond the bounds of the settlement. Remember, you do **not** need to write any code. [3]

.....

.....

.....

.....

.....

4. *AQA Burgers* decide to install a large warehouse at their main headquarters (500,500) and would like to calculate the best order for their delivery truck to visit the other outlets in so that the shortest possible distance is travelled.

   Note that there is a graph of all the outlets in the figure for Q6 which may help you to visualise the problem.

   a) State the Big O of this problem with *n* being the number of outlets. [1]

   .....

   b) This is an example of an intractable problem, state what is meant by this. [1]

   .....

   .....

   c) Describe and justify a heuristic method that could be used to reduce the time complexity of this problem so that it is no longer intractable. [2]

   .....

   .....

   .....

   .....

5. Describe in full how the `GetDistanceBetweenTwoOutlets` subroutine of the `Company` class calculates the distance between two outlets. [4]

.....

.....

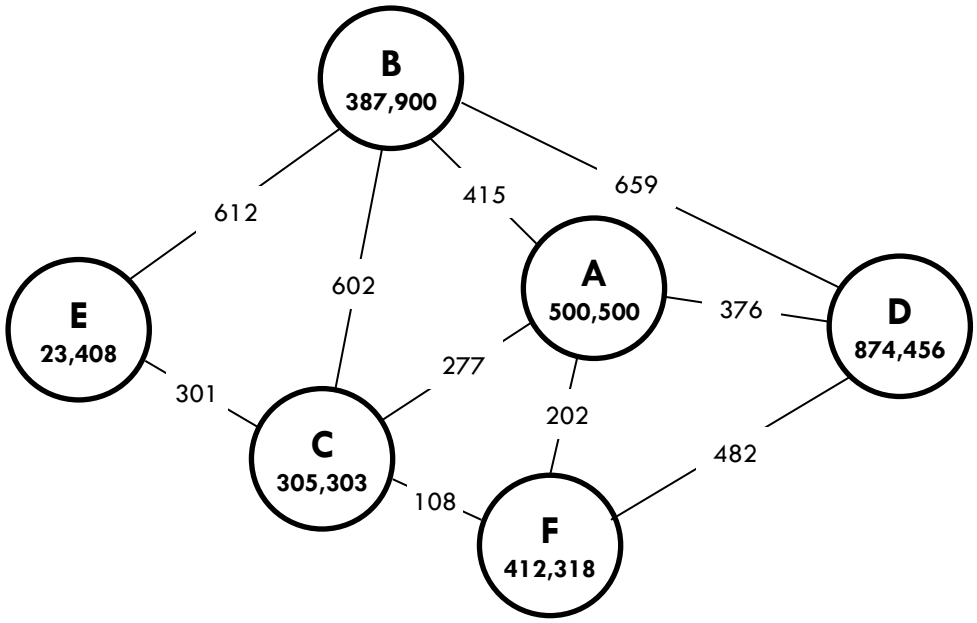
.....

.....

.....

.....

6. The distribution of outlets for the *AQA Burgers* restaurant chain can be represented using a graph such as the one shown here (not to scale):



An automated delivery network is being set up between *AQA Burgers'* outlets. As such, each outlet needs to know the shortest route to each other outlet so that deliveries can be optimised.

Describe the steps (in any form you like) for an algorithm which determines the shortest route from one outlet to another. Assume that the automated vehicles are restricted to the routes shown above.

Remember, you do **not** need to write any code. [6]

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

7. Explain the role of the variable `UpOrDown` in the `ProcessCostOfFuelChangeEvent` subroutine of the `Simulation` class. [3]

.....

.....

.....

.....

.....

8. In the `Simulation` constructor, the integer literals 100,000, 200 and 203 are passed to the `Company` constructor when creating the 'AQA Burgers' company. State the role of each of these integer literals. [3]

.....

.....

.....

.....

.....

9. Describe in full the operation of the `GetIndexOfCompany` subroutine in the `Simulation` class. [5]

.....

.....

.....

.....

.....

.....

.....

.....

10. Describe the circumstances under which the `ModifyCompany` subroutine of the `Simulation` class would output the text 'Invalid coordinates'. [3]

.....

.....

.....

.....

.....

11. Currently, a call to the `LargeSettlement` constructor could not result in a settlement that is smaller than 1,000 by 1,000. Explain how this could happen and describe how the constructor could be changed in order to prevent a settlement smaller than 1,000 by 1,000 from being created. You do **not** need to write any code. [3]

.....

.....

.....

.....

.....

12. Describe exactly how the `ProcessDayEnd` subroutine works in the `Simulation` class. [5]

.....

.....

.....

.....

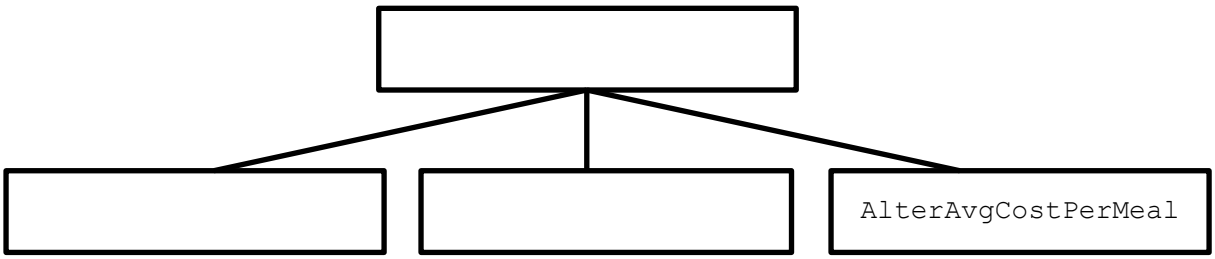
.....

.....

.....

.....

13. Complete the following hierarchy chart for part of the `Simulation` class of the Skeleton Program. You should **not** include calls to any library subroutines. [3]



14. Describe how the program would respond to a call to the `Company` constructor using a category that is neither 'fast food', 'family' nor 'named chef'. [2]

.....

.....

.....