# Food Magnate Simulation

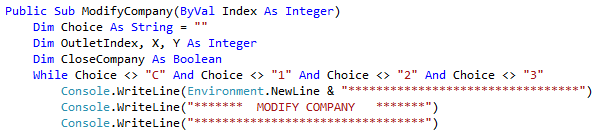
## Programming Tasks – Suggested Mark Scheme

Note that the following are recommended solutions, and not an exhaustive list of all possible solutions to each task. The marking guidance should be used as a guide only. Discretion should be used in awarding credit where alternative solutions are given.

## Task 1 (max. 6 marks)

**1 mark** loop set up in ModifyCompany to repeat until something other than 'C', '1', '2' or '3' is entered

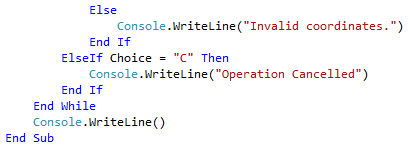
**1 mark** loop must contain call to menu display and user input, and must not begin with choice uninitialised



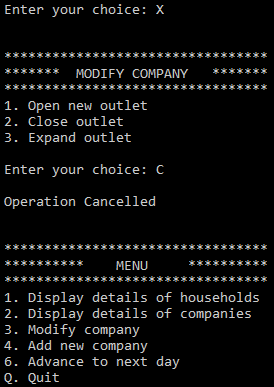
**1 mark** selection statement to catch entry of an upper-case 'C'

**1 mark** ‘operation cancelled’ displayed within selection clause

**1 mark** all inputs, old and new, dealt with correctly



**1 mark** screenshot showing 'X' causing a repeat of the loop and 'C' causing redisplay of main menu:



## Task 2 (max. 9 marks)

**1 mark** Boolean or equivalent to store whether a collision has occurred

**1 mark** outer loop attempts to continue until valid location is found, even if unsuccessful (intent must be clear)

**1 mark** random location generated inside outer loop

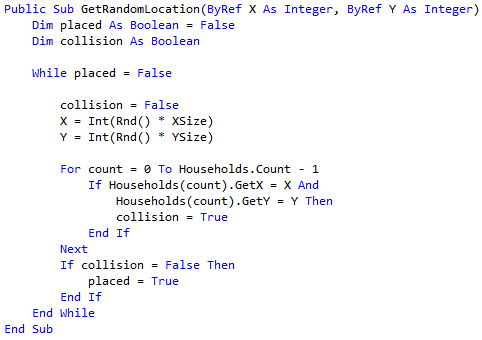
**1 mark** inner loop to iterate through Households list

**1 mark** selection statement to determine whether location is already occupied

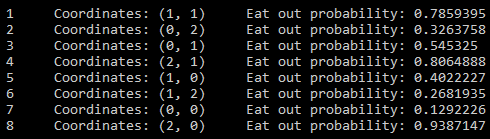
**1 mark** flag set in inner loop to store that a location was already occupied

**1 mark** Boolean (placed in this code) set to reflect no collision occurring

**1 mark** when outer loop ends, X and Y will always represent a non-colliding location



**1 mark** eight household locations are all different:



## Task 3 (max. 7 marks)

**1 mark** new subroutine declared correctly in the Outlet class

**1 mark** MaxCapacity multiplied by the parameter (name of parameter unimportant)

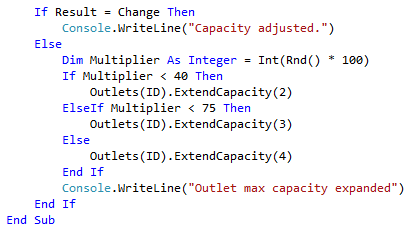


**1 mark** call to generate a random number that could be appropriate to the 40/35/25 distribution in ExpandOutlet

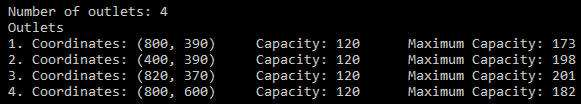
**1 mark** probabilities are actually 40%, 35% and 25%

**1 mark** correct parameter passed to ExtendCapacity in all circumstances

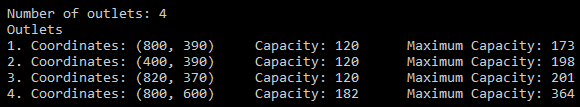
**1 mark** only output message in these circumstances is 'outlet max capacity expanded'



**1 mark** correct message output, and Paltry Poultry outlet 4 max capacity doubled, tripled or quadrupled:







## Task 4 (max. 8 marks)

**1 mark** new subroutine, correct name, type and parameters in the Settlement class

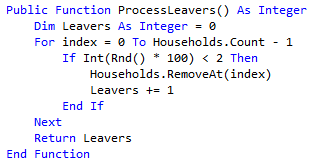
**1 mark** integer to store number of leavers

**1 mark** loop to iterate over all households

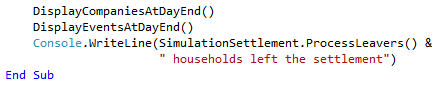
**1 mark** selection structure, based on 2% probability

**1 mark** removal of household from list, inside selection clause

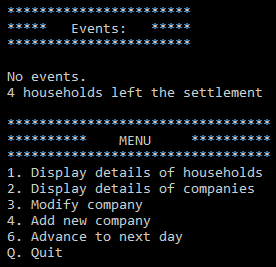
**1 mark** integer variable incremented inside selection clause, and returned after loop



**1 mark** call to ProcessLeavers as the final instruction in ProcessDayEnd



**1 mark** screenshot displays number of households removed, which might be the number by itself (i.e. without the text 'households left the settlement') and/or zero:



## Task 5 (max. 7 marks)

**1 mark** new input message within code executed when user enters '3'

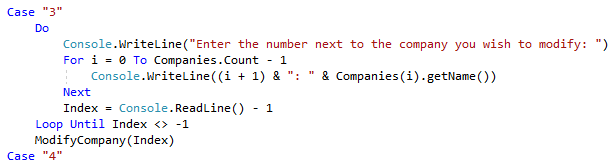
**1 mark** loop to iterate over all companies in the list

**1 mark** output to contain call to company's GetName subroutine

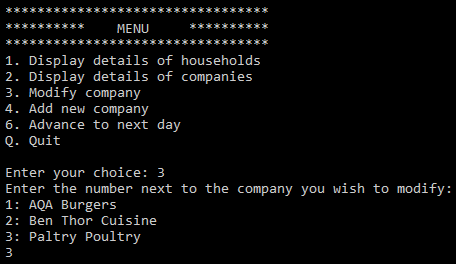
**1 mark** output to display 1-based indices instead of 0-based indices

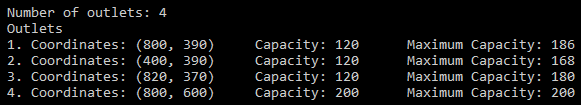
**1 mark** user input stored in the index variable

**1 mark** user input decremented

****

**1 mark** user input should now be based on a list, with outlets numbered 1, 2, 3; final outlet of 'Paltry Poultry' should have a capacity equal to its max capacity:



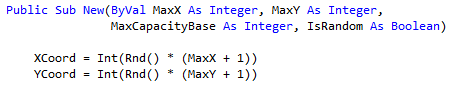


## Task 6 (max. 5 marks)

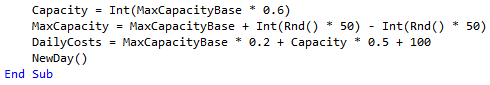
**1 mark** subroutine correctly declared as a constructor

**1 mark** correct parameters declared

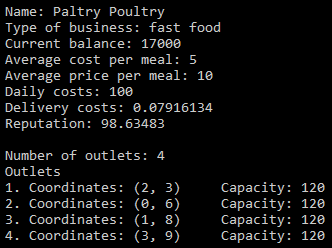
**1 mark** XCoord and YCoord set to random integers between 0 and MaxX/MaxY inclusive



**1 mark** the rest of the subroutine executes as normal. **A.** call to a new subroutine (to which existing constructor might call) that contains the extra lines. **R.** if call is made to other constructor.

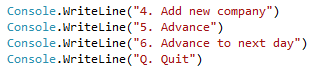
****

**1 mark** screen capture shows values for outlet coordinates that are all 10 or lower:



## Task 7 (max. 7 marks)

**1 mark** option 5 correctly added in DisplayMenu



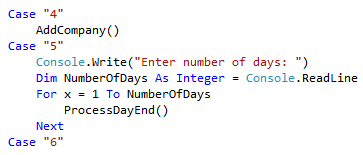
**1 mark** selection structure in Run captures an input of 5

**1 mark** any suitable prompt for the user to enter a number of days

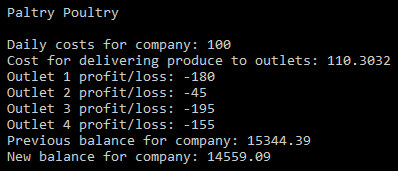
**1 mark** integer variable set to user input

**1 mark** loop that runs the correct number of times

**1 mark** call to ProcessDayEnd inside the loop and no other code



**1 mark** three days' worth of financials and events should be displayed, although if the previous balance for Paltry Poultry is anything other than 17000, and would match code, credit can be given:



## Task 8 (max. 9 marks)

**1 mark** prompt updated to include reference to '4' generating a random company

**1 mark** loop updated to include '4' as a terminating condition



**1 mark** updating selection structure to ensure 3 results in 'named chef'

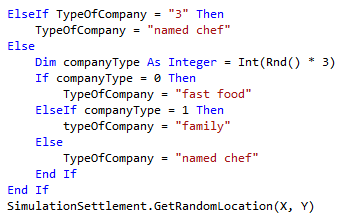
**1 mark** addition to selection structure to catch 4 or 'else' (i.e. not 1, 2 or 3)

**1 mark** random number generated, even if likelihoods are not evenly distributed

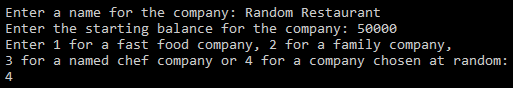
**1 mark** selection structure sets restaurant type according to random number

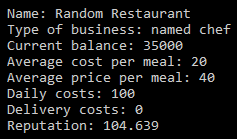
**1 mark** three company types are equally likely to be created

**1 mark** additional code does not impede code from the Skeleton Program



**1 mark** input of '4', 'Random Restaurant' and '50000', resulting in a company of any type being created



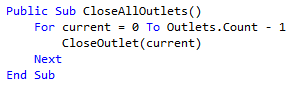


## Task 9 (max. 6 marks)

**1 mark** new subroutine created, with no parameters and no return

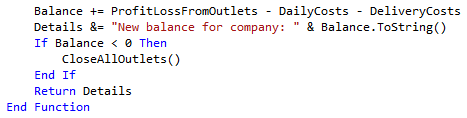
**1 mark** loop to iterate over all outlets in the Outlets list

**1 mark** call to CloseOutlet for each outlet

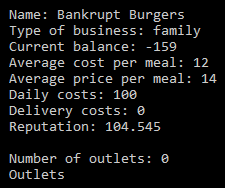


**1 mark** selection statement in ProcessDayEnd to check for balance of less than zero, immediately before return statement

**1 mark** call to CloseAllOutlets in selection structure



**1 mark** entering '2' in the main menu should reveal that 'Bankrupt Burgers' has no outlets:



## Task 10 (max. 11 marks)

**1 mark** class definition, which includes inheritance

**1 mark** constructor correctly declared, with two parameters

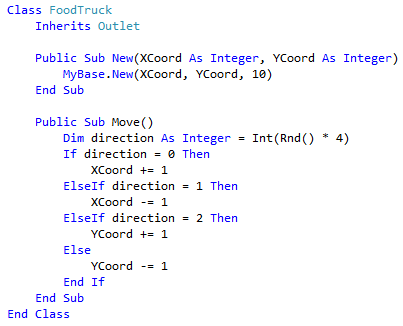
**1 mark** call within constructor to the Outlet constructor, passing correct values

**1 mark** Move subroutine declared

**1 mark** random number generated

**1 mark** selection statement uses four different possible values of random number

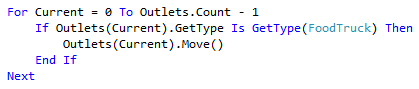
**1 mark** each of north, south, east and west correctly simulated

****

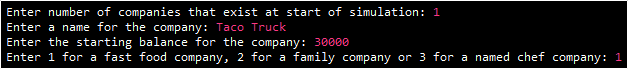
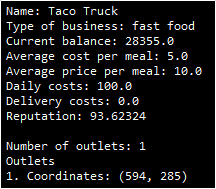
**1 mark** either creation of loop to iterate through outlets in ProcessDayEnd subroutine, or an attempt to use the existing loop to call move, even if syntactically invalid

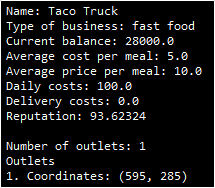
**1 mark** selection statement to check whether an outlet is an instance of a FoodTruck

**1 mark** move subroutine called for all FoodTruck objects and only FoodTruck objects

****

**1 mark** screen captures show a difference of 1 in **either** X **or** Y coordinates between days:

****

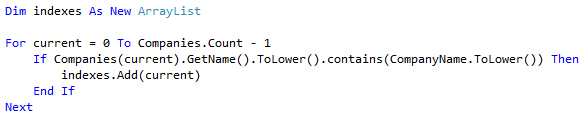
****

## Task 11 (max. 15 marks)

**1 mark** data structure created to store indexes

**1 mark** selection structure has been changed from 'equals' to 'contains' or equivalent

**1 mark** inside selection structure, adding the index to the data structure



**1 mark** selection structure to check for only a single match, **A.** if 'else' by process of elimination

**1 mark** correct index returned for a single match

****

**1 mark** selection structure to check for multiple matches, **A.** if 'else' by process of elimination

**1 mark** prompt for user entry, either before or after attempt to display matches

**1 mark** loop to iterate through all matches in an attempt to display them

**1 mark** name of each matching outlet displayed

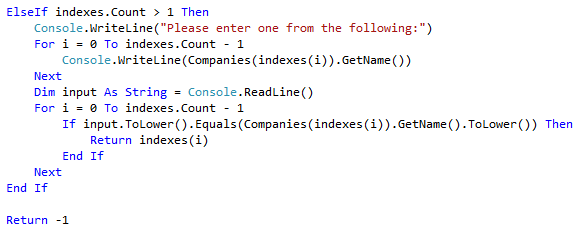
**1 mark** user input requested only if multiple matches have been found

**1 mark** loop to iterate through the matches in an attempt to compare with user input

**1 mark** comparison ('equals', not 'contains') is made inside the loop, with case ignored

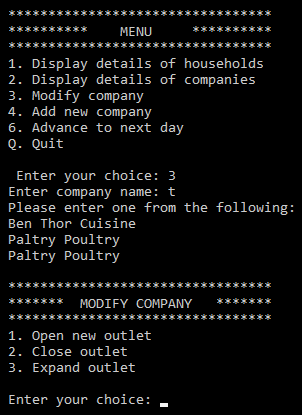
**1 mark** correct index returned in the event of a match

**1 mark** value of -1 returned if no matches are found or if second entry does not match



***(continues on next page)***

**1 mark** screen capture displays 'Paltry Poultry' and 'Ben Thor Cuisine', with 'Paltry Poultry' entered and accepted:



## Task 12 (max. 7 marks)

**1 mark** outlet with a matching ID is accessed

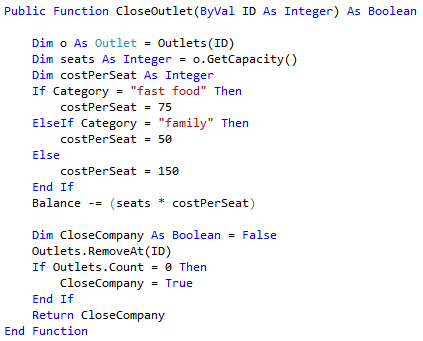
**1 mark** number of seats is set to a call to GetCapacity

**1 mark** selection structure uses Category attribute of the Company class

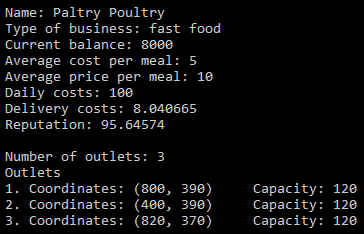
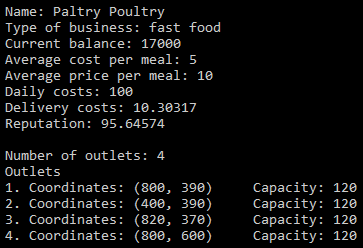
**1 mark** cost per seat is set correctly for all categories

**1 mark** balance is decremented by the cost per seat multiplied by the number of seats

**1 mark** the original code, to remove the outlet and check for zero outlets, should run unchanged

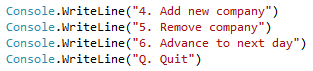


**1 mark** current balance of Paltry Poultry should change from 17000 to 8000, and there should be one outlet fewer:

****

## Task 13 (max. 8 marks)

**1 mark** additional line added to DisplayMenu with correct number and text



**1 mark** addition of '5' to the selection structure in run

**1 mark** prompt for company name and storage of user input in a string variable

**1 mark** selection structure to check whether the index either is or is not -1

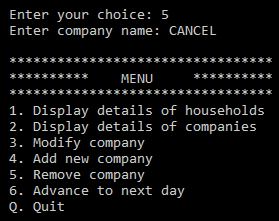
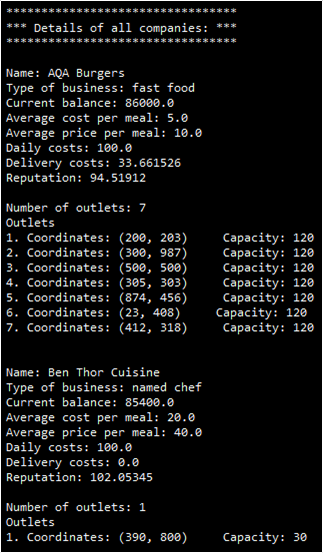
**1 mark** removal of company at the correct index, only if the index is not -1

**1 mark** loop will terminate if the loop is not -1

**1 mark** loop will terminate if 'cancel', in any combination of upper case / lower case, is entered



**1 mark** entering 'CANCEL' returns user to the main menu; entering 'Paltry Poultry' results in only two companies being present in the simulation – AQA Burgers and Ben Thor Cuisine:



## Task 14 (max. 20 marks)

**1 mark** GetBalance declared in the Company class, with no parameters and correct type

**1 mark** correct return statement, **R.** if any additional code

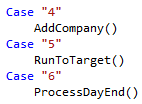


**1 mark** additional option added to DisplayMenu



**1 mark** selection structure modified to include '5'

**1 mark** call to new RunToTarget subroutine if '5' is entered



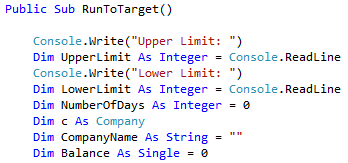
**1 mark** new RunToTarget subroutine declared

**1 mark** user prompted for an upper limit and a lower limit

**1 mark** each user input stored as a separate integer

**1 mark** variable names UpperLimit and LowerLimit used as instructed

**1 mark** variables for number of days, balance and company name, of appropriate types, declared



**1 mark** loop to run until a balance is equal to or above the upper limit, or equal to or below the lower limit

**1 mark** loop to iterate over each company in the simulation

**1 mark** comparison with both upper and lower limits

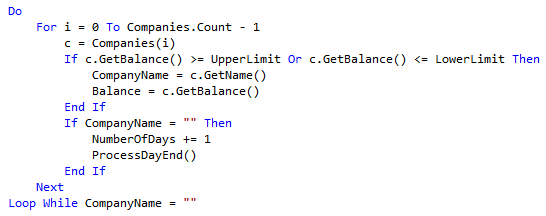
**1 mark** call to ProcessDayEnd once within each iteration

**1 mark** no call to ProcessDayEnd if a balance has already reached termination condition, including if the termination condition was already reached by the start of the first loop (i.e. zero days should be a possibility)

**1 mark** number of days incremented within each iteration

***(continues on next page)***

**1 mark** calls to GetName and GetBalance have occurred before loop terminates

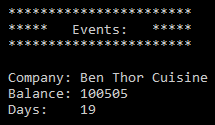


**1 mark** values for the company's name and balance, and the number of days, are output

**1 mark** output values are correct under all circumstances



**1 mark** screen evidence showing that multiple days have passed, and the balance is either >= 100000 or <= 0:



## Task 15 (max. 14 marks)

**1 mark** class definition with correct identifier, which inherits from LargeSettlement

**1 mark** valid constructor with three integer parameters

**1 mark** valid call within constructor to constructor of superclass

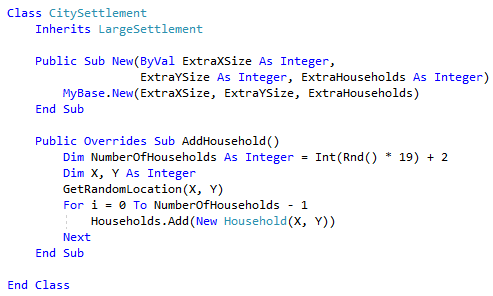
**1 mark** subroutine AddHousehold declared with Overrides modifier

**1 mark** generation of random number, within AddHousehold, between 2 and 20 inclusive

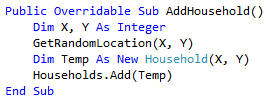
**1 mark** generation of random X and Y coordinates within the settlement (easiest via a call to the Settlement class's GetRandomLocation subroutine, but any valid approach can be credited)

**1 mark** loop set up that will iterate once for each household in this location (integer between 2 and 20)

**1 mark** each new household, within the loop, added to the Households list



**1 mark** Overridable modifier added to pre-existing AddHouseholds subroutine



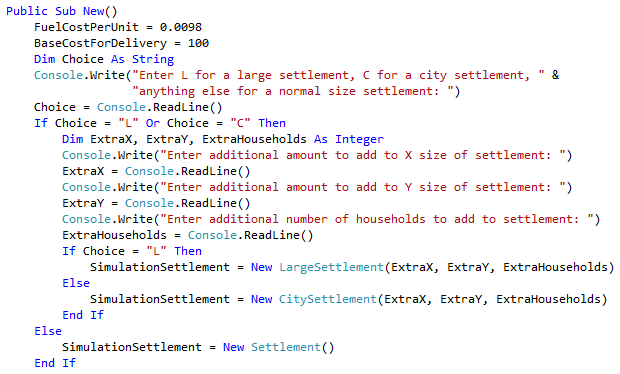
***(continues on next page)***

**1 mark** first prompt of the simulation amended to allow user to select a city settlement

**1 mark** user selecting a city settlement results in prompts for additional X and Y coordinates and number of households

**1 mark** user selecting a city settlement results in call to CitySettlement constructor

**1 mark** other inputs should continue to work as previously (i.e. new code does not disrupt old code)



**1 mark** screen capture should show consecutive households at the same location

