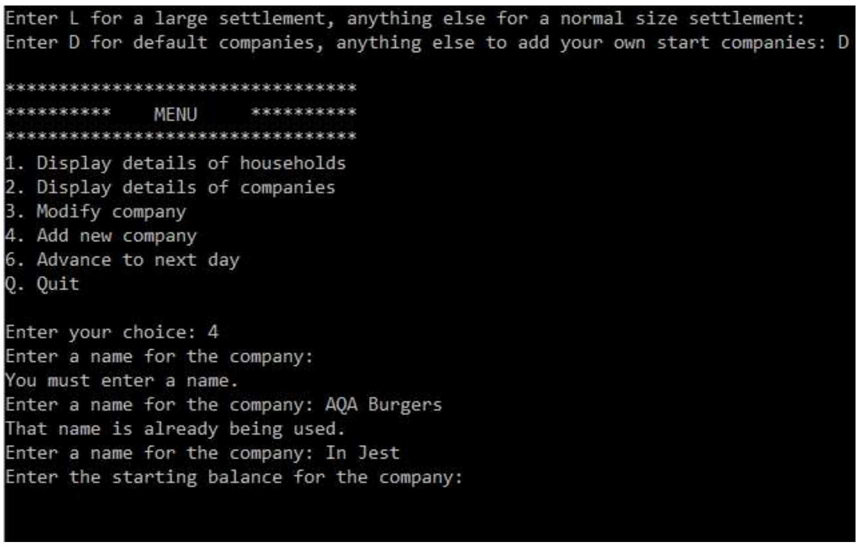
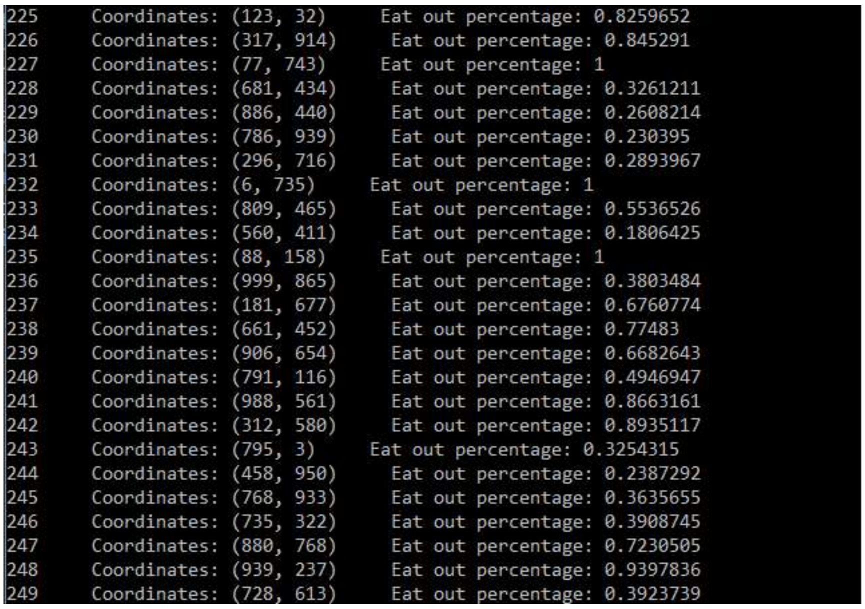
Question			Marks
07	1	**Mark is for AO2 (analyse)** `GetDetails`; **R.** if spelt incorrectly **R.** if any additional code **I.** case and spacing	1
07	2	**Mark is for AO2 (analyse)** `Details`; `OldCapacity`; **R.** if spelt incorrectly **R.** if any additional code **I.** case and spacing **Max 1 mark**	1
07	3	**Mark is for AO1 (understanding)** Private attributes can only be accessed by the class/object they belong to whereas protected attributes can also be accessed by any classes that inherit from the class they belong to; **A.** file instead of class/object (Java only) **NE.** private attribute can only be accessed by the class/object they belong to whereas protected attributes can be accessed by others classes/objects	1
07	4	**Mark is for AO2 (analyse)** The calculation of the daily costs will be inconsistent as it will be different in the `AlterCapacity` method; **A.** it won't have been updated in other locations which use that constant	1

Question			Marks
08	1	**Mark is for AO2 (analyse)** `LargeSettlement`; **R.** if spelt incorrectly **R.** if any additional code **I.** case and spacing	1
08	2	**Mark is for AO2 (analyse)** `Household`; **R.** if spelt incorrectly **R.** if any additional code **I.** case and spacing	1
08	3	**Mark is for AO2 (analyse)** `Company`; **R.** if spelt incorrectly **R.** if any additional code **I.** case and spacing	1
08	4	**Mark is for AO2 (analyse)** Aggregation;	1

Question			Marks
09		**All marks are for AO2 (analyse)** It stores the cumulative reputation for the companies in an array/list; It then generates a random number which is less than the total reputation; **A.** generates a random number based on the total reputation of the companies Finds the first cumulative reputation that the number is less than; The position of this cumulative reputation in the list indicates the company that the household will use;	4

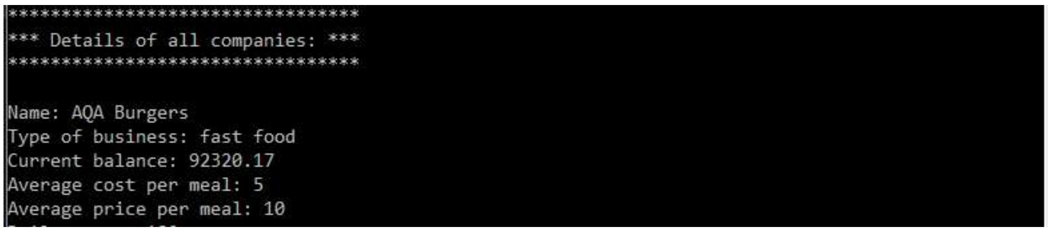
Question			Marks
10	1	**All marks for AO3 (programming)** 1. Indefinite iterative structure contains code that gets the name from the user; 2. One correct condition; 3. Both correct conditions and correct logic for the iterative structure; 4. Displays error message if no name is entered // displays error message if a name that has already been used is entered; 5. Displays error message under all correct circumstances and only under correct circumstances; **Max 4** if code contains errors	5
10	2	**Mark is for AO3 (evaluate)** ****** SCREEN CAPTURE ****** *Must match code from **10.1**, including prompts on screen capture matching those in code.* *Code for **10.1** must be sensible.* Screen captures showing error message(s) being shown for the two invalid names and then showing the message asking for the starting balance when a valid name is entered; 	1

```
Enter L for a large settlement, anything else for a normal size settlement:
Enter D for default companies, anything else to add your own start companies: D

*********************************
**********    MENU     **********
*********************************
1. Display details of households
2. Display details of companies
3. Modify company
4. Add new company
6. Advance to next day
Q. Quit

Enter your choice: 4
Enter a name for the company:
You must enter a name.
Enter a name for the company: AQA Burgers
That name is already being used.
Enter a name for the company: In Jest
Enter the starting balance for the company:
```

Question			Marks
11	1	**All marks for AO3 (programming)** 1. Creating a new class called `AffluentHousehold`; **R.** other names for class **I.** case and minor typos 2. New class inherits from `Household`; 3. Constructor created that overrides base class constructor with call made to base class constructor; **R.** if incorrect parameters 4. Sets the value of `ChanceEatOutPerDay` to 1; **R.** if before call to base class constructor **R.** If not after attempt at call to base class constructor The following all relate to the `AddHousehold` method: 5. Selection structure with correct condition; 6. Creates an `AffluentHousehold` object; **R.** if it also creates a household 7. Creates an `AffluentHousehold` under the correct circumstances and a `Household` under the correct circumstances; **R.** if new household not added to `Households` **Max 6** if code contains errors	7
11	2	**Mark is for AO3 (evaluate)** ****** SCREEN CAPTURE ****** *Must match code from **11.1**, including prompts on screen capture matching those in code.* *Code for **11.1** must be sensible.* Screen capture(s) showing that households with an X value less than 100 have an eat out percentage of 1; 	1

```
225     Coordinates: (123, 32)     Eat out percentage: 0.8259652
226     Coordinates: (317, 914)     Eat out percentage: 0.845291
227     Coordinates: (77, 743)     Eat out percentage: 1
228     Coordinates: (681, 434)     Eat out percentage: 0.3261211
229     Coordinates: (886, 440)     Eat out percentage: 0.2608214
230     Coordinates: (786, 939)     Eat out percentage: 0.230395
231     Coordinates: (296, 716)     Eat out percentage: 0.2893967
232     Coordinates: (6, 735)     Eat out percentage: 1
233     Coordinates: (809, 465)     Eat out percentage: 0.5536526
234     Coordinates: (560, 411)     Eat out percentage: 0.1806425
235     Coordinates: (88, 158)     Eat out percentage: 1
236     Coordinates: (999, 865)     Eat out percentage: 0.3803484
237     Coordinates: (181, 677)     Eat out percentage: 0.6760774
238     Coordinates: (661, 452)     Eat out percentage: 0.77483
239     Coordinates: (906, 654)     Eat out percentage: 0.6682643
240     Coordinates: (791, 116)     Eat out percentage: 0.4946947
241     Coordinates: (988, 561)     Eat out percentage: 0.8663161
242     Coordinates: (312, 580)     Eat out percentage: 0.8935117
243     Coordinates: (795, 3)     Eat out percentage: 0.3254315
244     Coordinates: (458, 950)     Eat out percentage: 0.2387292
245     Coordinates: (768, 933)     Eat out percentage: 0.3635655
246     Coordinates: (735, 322)     Eat out percentage: 0.3908745
247     Coordinates: (880, 768)     Eat out percentage: 0.7230505
248     Coordinates: (939, 237)     Eat out percentage: 0.9397836
249     Coordinates: (728, 613)     Eat out percentage: 0.3923739
```

Question			Marks
12	1	**All marks for AO3 (programming)** Marks for changes to the `Simulation` class: 1. Two extra options displayed on the modify company menu using appropriate messages; 2. Selection structures for the new menu options with appropriate condition(s); 3. Gets the user to enter the interest rate when getting a loan and the amount to pay back when paying back under the appropriate circumstances; **A.** done in appropriate places in the `Company` class; 4. Calls to appropriate methods in `Company` class in the selection structures; Marks for changes to the `Company` class: 5. Attributes of appropriate data types created for `LoanBalance` and `InterestRate`; 6. Correct calculation of daily interest payment and new balance in `ProcessDayEnd`; **R.** if the balance is changed before previous balance concatenated with `Details` 7. Selection structure to check if `LoanBalance` is 0 when user chooses to get a loan; **A.** check for less than or equal to 0 8. `Balance`, `LoanBalance` and `InterestRate` set to correct values in the selection structure; 9. `LoanBalance` and `Balance` changed by the correct amount when user chooses to pay back part of the loan; 10. All attributes in `Company` are only accessed and modified by methods in `Company`; **R.** if no attempt to access or modify the attributes used when getting or paying back a loan. **Max 9 marks** if code contains errors	10

Question			Marks
12	2	**Mark is for AO3 (evaluate)** **** **SCREEN CAPTURE** **** *Must match code from **12.1**, including prompts on screen capture matching those in code.* *Code for **12.1** must be sensible.* Screen capture(s) showing that the balance for AQA Burgers is approximately 92 000; **Note for examiners:** due to random numbers in simulation exact balance can vary. 	1

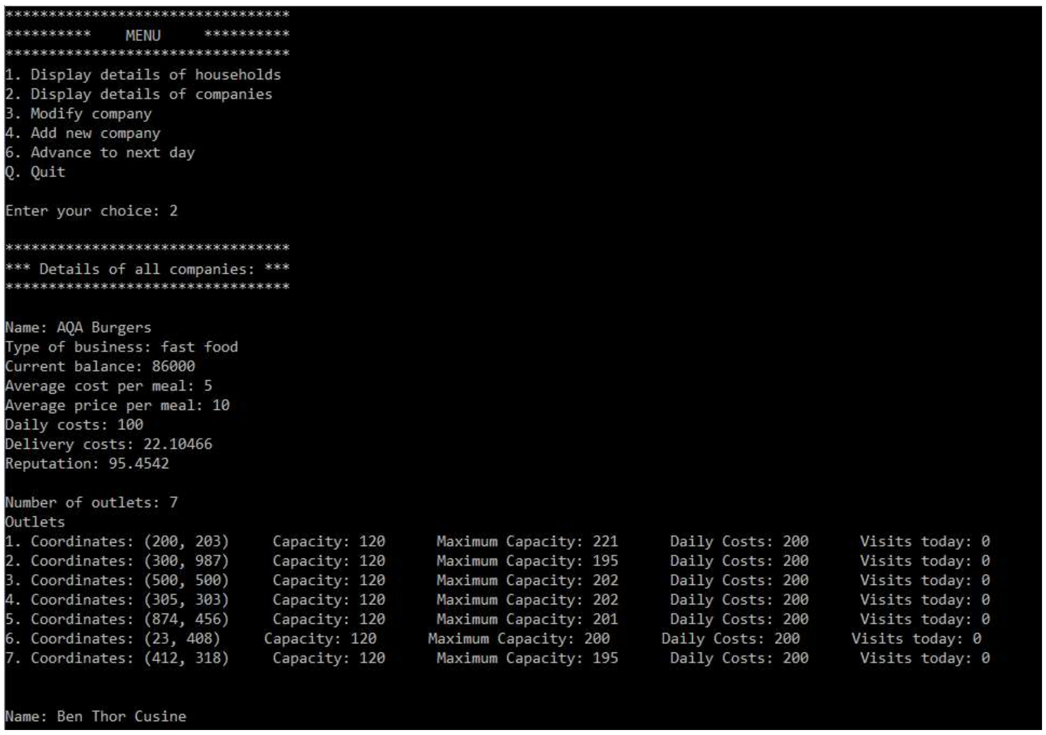
```
*********************************
*** Details of all companies: ***
*********************************

Name: AQA Burgers
Type of business: fast food
Current balance: 92320.17
Average cost per meal: 5
Average price per meal: 10
```

Question			Marks
13	1	**All marks for AO3 (programming)** 1. Created new method called `GetOrderedListOfOutlets`; **R.** other names for method **I.** case and minor typos 2. Method returns a list/array; 3. Outlet 0 is added to the route first; 4. Iterative structure that repeats until all outlets have been added to the route; 5. Has variable that is used to store shortest distance found between two nodes so far and a variable to store which outlet results in the shortest distance; 6. Iterative structure that looks at each outlet for which distance from previous outlet in route needs to be calculated; **A.** looks at all outlet except previous outlet 7. No outlet can appear more than once in route created; **R.** if adds or two or fewer outlets to the list only **R.** if no attempt to check if outlet has already been added or equivalent 8. Route created contains all the company's outlets; 9. Shortest distance between two nodes variable set to suitable starting value and reset after each outlet (except last one) is added to route; 10. `GetOrderedListOfOutlets` implements the algorithm described in **Figure 6** in the question; 11. Modified `CalculateDeliveryCost` so that it calls `GetOrderedListOfOutlets` instead of `GetListOfOutlets`; **A.** alternative identifier used as long as match that used for mark point 1 **Max 10** if code contains errors or if other parts of the subroutine no longer work correctly	11

Question			Marks
13	2	**Mark is for AO3 (evaluate)**	1

****** SCREEN CAPTURE ******
*Must match code from **13.1**, including prompts on screen capture matching those in code.*
*Code for **13.1** must be sensible.*

Screen capture(s) showing that the delivery cost for AQA Burgers is 22.10446;



```
*********************************
**********    MENU     **********
*********************************
1. Display details of households
2. Display details of companies
3. Modify company
4. Add new company
6. Advance to next day
Q. Quit

Enter your choice: 2

*********************************
*** Details of all companies: ***
*********************************

Name: AQA Burgers
Type of business: fast food
Current balance: 86000
Average cost per meal: 5
Average price per meal: 10
Daily costs: 100
Delivery costs: 22.10466
Reputation: 95.4542

Number of outlets: 7
Outlets
1. Coordinates: (200, 203)     Capacity: 120      Maximum Capacity: 221      Daily Costs: 200      Visits today: 0
2. Coordinates: (300, 987)     Capacity: 120      Maximum Capacity: 195      Daily Costs: 200      Visits today: 0
3. Coordinates: (500, 500)     Capacity: 120      Maximum Capacity: 202      Daily Costs: 200      Visits today: 0
4. Coordinates: (305, 303)     Capacity: 120      Maximum Capacity: 202      Daily Costs: 200      Visits today: 0
5. Coordinates: (874, 456)     Capacity: 120      Maximum Capacity: 201      Daily Costs: 200      Visits today: 0
6. Coordinates: (23, 408)     Capacity: 120      Maximum Capacity: 200      Daily Costs: 200      Visits today: 0
7. Coordinates: (412, 318)     Capacity: 120      Maximum Capacity: 195      Daily Costs: 200      Visits today: 0


Name: Ben Thor Cusine
```

C#

Question			Marks
05	1	(code below)	12

```
int current;
int[] frequencies = new int[10];
int modeFrequency = 0;
bool multimodal = false;
int noOfDigits;
Console.Write("Enter number of digits: ");
noOfDigits = Convert.ToInt32(Console.ReadLine());
for (int i = 0; i < noOfDigits; i++)
{
    Console.Write("Enter a numeric digit: ");
    current = Convert.ToInt32(Console.ReadLine());
    frequencies[current] += 1;
}
for (int i = 0; i < 10; i++)
{
    if (frequencies[i] > modeFrequency)
    {
        modeFrequency = frequencies[i];
        multimodal = false;
    }
    else if (frequencies[i] == modeFrequency )
    {
        multimodal = true;
    }
}
if (multimodal)
{
    Console.WriteLine("Data was multimodal");
}
else
{
    Console.WriteLine("The modal digit appeared " + modeFrequency +
" times");
}
Console.ReadLine();
```

Question			Marks
10	1	(code below)	5

```
private void AddCompany()
{
    int balance, x = 0, y = 0;
    string companyName, typeOfCompany = "9";
    do
    {
        Console.Write("Enter a name for the company: ");
        companyName = Console.ReadLine();
        if (companyName == "")
        {
            Console.WriteLine("You must enter a name.");
        }
        else if (GetIndexOfCompany(companyName) != -1)
        {
            Console.WriteLine("That name is already being used.");
        }
    } while (companyName == "" || GetIndexOfCompany(companyName) !=
-1);
Console.Write("Enter the starting balance for the company: ");
```

Alternative answer

```
private void AddCompany()
{
    int balance, x = 0, y = 0;
    string companyName, typeOfCompany = "9";
    Console.Write("Enter a name for the company: ");
    companyName = Console.ReadLine();
    while (companyName == "" || GetIndexOfCompany(companyName) != -
1)
    {
        if (companyName == "")
        {
            Console.WriteLine("You must enter a name.");
        }
        else if (GetIndexOfCompany(companyName) != -1)
        {
            Console.WriteLine("That name is already being used.");
        }
        Console.Write("Enter a name for the company: ");
        companyName = Console.ReadLine();
    }
    Console.Write("Enter the starting balance for the company: ");
```

11 1 — **7**

```
public void AddHousehold()
{
    int x = 0, y = 0;
    GetRandomLocation(ref x, ref y);
    if (x < 100)
    {
        AffluentHousehold temp = new AffluentHousehold(x, y);
        households.Add(temp);
    }
    else
    {
        Household temp = new Household(x, y);
        households.Add(temp);
    }
}

class AffluentHousehold : Household
{
    public AffluentHousehold(int x, int y)
        :base(x, y)
    {
        chanceEatOutPerDay = 1;
    }
}
```

12 1 — **10**

From the simulation class

```
public void ModifyCompany(int index)
{
    string choice;
    int outletIndex, x, y;
    bool closeCompany;
    Console.WriteLine("\n*********************************");
    Console.WriteLine("*******  MODIFY COMPANY   *******");
    Console.WriteLine("*********************************");
    Console.WriteLine("1. Open new outlet");
    Console.WriteLine("2. Close outlet");
    Console.WriteLine("3. Expand outlet");
    Console.WriteLine("4. Get Loan");
    Console.WriteLine("5. Pay back loan");
    Console.Write("\nEnter your choice: ");
    choice = Console.ReadLine();
    if (choice == "2" || choice == "3")
    ...
    else if (choice == "1")
    ...
    else if (choice == "4")
    {
        Console.Write("Enter the interest rate for the loan: ");
        double rate = Convert.ToDouble(Console.ReadLine());
        companies[index].GetLoan(rate);

    }
    else if (choice == "5")
    {
        Console.Write("Enter the amount to pay back: ");
        double payBackAmount = Convert.ToDouble(Console.ReadLine());
        companies[index].PayBackLoan(payBackAmount);
    }
    Console.WriteLine();
}
```

From the company class

```
class Company
{
    private static Random rnd = new Random();
    protected string name, category;
    protected double balance, reputationScore, avgCostPerMeal,
avgPricePerMeal, dailyCosts, familyOutletCost, fastFoodOutletCost,
namedChefOutletCost, fuelCostPerUnit, baseCostOfDelivery;
    protected List<Outlet> outlets = new List<Outlet>();
    protected int familyFoodOutletCapacity, fastFoodOutletCapacity,
namedChefOutletCapacity;
    protected double loanBalance;
    protected double interestRate;
...

public string ProcessDayEnd()
...
    }
    details += "Previous balance for company: " + balance.ToString()
+ "\n";
    balance += profitLossFromOutlets - dailyCosts - deliveryCosts -
(loanBalance * interestRate);
    details += "New balance for company: " + balance.ToString();
    return details;
}

public void GetLoan(double rate)
{
    if (loanBalance == 0)
    {
        balance += 10000;
        interestRate = rate;
    }
}

public void PayBackLoan(double amount)
{
    loanBalance -= amount;
    balance -= amount;
}
```

Alternative answer for taking a loan

In `ModifyCompany` method in `Simulation` class

```
...
else if (choice == "4")
                {
                    if (companies[index].GetLoanBalance() <= 0)
                    {
                        Console.Write("Enter the interest rate for
the loan: ");
                        double rate =
Convert.ToDouble(Console.ReadLine());
                        companies[index].TakeOutLoan(rate);
                    }
                }
```

Methods in `Company` Class

```
public double GetLoanBalance()
{
    return loanBalance;
}

public void TakeOutLoan(double interestRate)
{
    this.interestRate = interestRate;
    loanBalance = 10000;
    balance += 10000;
}
```

13 **1** — **11**

```
private List<int> GetOrderedListOfOutlets()
{
    List<int> orderedList = new List<int>();
    int nearestOutlet = 0; ;
    orderedList.Add(0);
    while (orderedList.Count < outlets.Count)
    {
        double shortestDistanceSoFar = 1000000;
        for (int count = 1; count < outlets.Count ; count++)
        {
            if (!orderedList.Contains(count))
            {
                double temp =
GetDistanceBetweenTwoOutlets(orderedList[orderedList.Count - 1],
count);
```

```
                if (temp < shortestDistanceSoFar)
                {
                    nearestOutlet = count;
                    shortestDistanceSoFar = temp;
                }
            }
        }
        orderedList.Add(nearestOutlet);
    }
    return orderedList;
}

public double CalculateDeliveryCost()
{
    List<int> listOfOutlets = new
List<int>(GetOrderedListOfOutlets());
    double totalDistance = 0;
    double totalCost = 0;
    for (int current = 0; current < listOfOutlets.Count - 1;
current++)
    {
        totalDistance +=
GetDistanceBetweenTwoOutlets(listOfOutlets[current],
listOfOutlets[current + 1]);
    }
    totalCost = totalDistance * fuelCostPerUnit;
    return totalCost;
}
```