

COIS 4470H : Modelling and Simulation : Assignment 1

Punyaja Mishra - 0660001

All questions are written in different computer programming languages to get comfortable with writing simulation programs in each language (C#, Python).

Question 1 : Situation

The program is written in C#. Since the probability are a percentage (1%, 2%, 3%), therefore, the random number is generated between [1,100] and then if the random number is equal to 1 or 0, that means, the server failed to boot up.

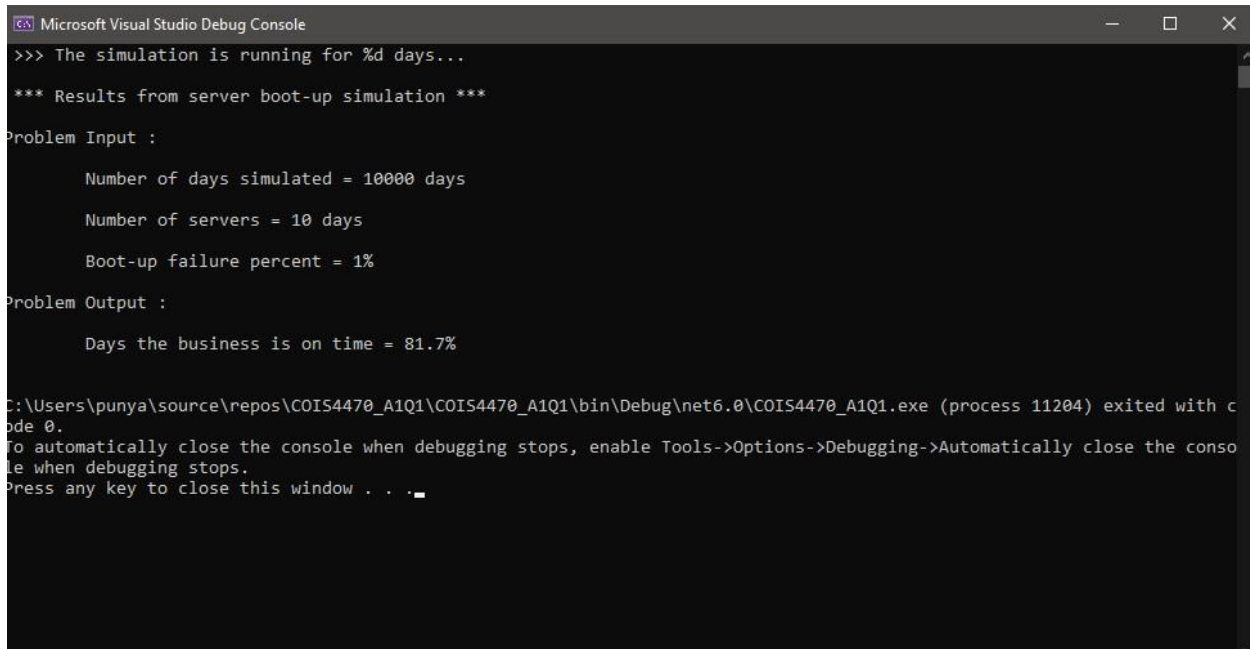
Case 1 : No Backup Server

Running the Program Simulation for FAIL_PERCENT = 1, 2, and 3.

Probability of Server failing to boot-up	1%	2%	3%
Percentage of days business on time	81.7%	74.08%	67.04%

Example output screenshot :

Fail Percent = 1%



```
Microsoft Visual Studio Debug Console
>>> The simulation is running for %d days...

*** Results from server boot-up simulation ***

Problem Input :

    Number of days simulated = 10000 days
    Number of servers = 10 days
    Boot-up failure percent = 1%

Problem Output :

    Days the business is on time = 81.7%

C:\Users\punya\source\repos\COIS4470_A1Q1\COIS4470_A1Q1\bin\Debug\net6.0\COIS4470_A1Q1.exe (process 11204) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Fail Percent = 2%

```
Microsoft Visual Studio Debug Console
>>> The simulation is running for %d days...

*** Results from server boot-up simulation ***

Problem Input :

    Number of days simulated = 10000 days
    Number of servers = 10 days
    Boot-up failure percent = 2%

Problem Output :

    Days the business is on time = 74.08%

C:\Users\punya\source\repos\COIS4470_A1Q1\COIS4470_A1Q1\bin\Debug\net6.0\COIS4470_A1Q1.exe (process 18468) exited with c
ode 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Fail Percent = 3%

```
Microsoft Visual Studio Debug Console
>>> The simulation is running for %d days...

*** Results from server boot-up simulation ***

Problem Input :

    Number of days simulated = 10000 days
    Number of servers = 10 days
    Boot-up failure percent = 3%

Problem Output :

    Days the business is on time = 67.04%

C:\Users\punya\source\repos\COIS4470_A1Q1\COIS4470_A1Q1\bin\Debug\net6.0\COIS4470_A1Q1.exe (process 12332) exited with c
ode 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Case 2 : 1 Backup Server

In this scenario, a new variable was added that keeps track of the number of servers failed on a particular day. If the number of servers failed are greater than 1, only then is the “sum_fail” incremented.

Running the Program Simulation for FAIL_PERCENT = 1, 2, and 3.

Probability of Server failing to boot-up	1%	2%	3%
Percentage of days business on time	98.37%	96.68%	94.23%

Example output screenshot:

Fail Percent = 1%

```

Microsoft Visual Studio Debug Console
>>> The simulation is running for %d days...

*** Results from server boot-up simulation ***

Problem Input :

    Number of days simulated = 10000 days

    Number of servers = 10 days

    Boot-up failure percent = 1%

Problem Output :

    Days the business is on time = 98.37%

C:\Users\punya\source\repos\COIS4470_A1Q1\COIS4470_A1Q1\bin\Debug\net6.0\COIS4470_A1Q1.exe (process 21868) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```

Fail Percent = 2%

```

Microsoft Visual Studio Debug Console
>>> The simulation is running for %d days...

*** Results from server boot-up simulation ***

Problem Input :

    Number of days simulated = 10000 days

    Number of servers = 10 days

    Boot-up failure percent = 2%

Problem Output :

    Days the business is on time = 96.68%

C:\Users\punya\source\repos\COIS4470_A1Q1\COIS4470_A1Q1\bin\Debug\net6.0\COIS4470_A1Q1.exe (process 10056) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```

Fail Percent = 3%

```
Microsoft Visual Studio Debug Console
>>> The simulation is running for %d days...

*** Results from server boot-up simulation ***

Problem Input :

    Number of days simulated = 10000 days
    Number of servers = 10 days
    Boot-up failure percent = 3%

Problem Output :

    Days the business is on time = 94.23%

C:\Users\punya\source\repos\COIS4470_A1Q1\COIS4470_A1Q1\bin\Debug\net6.0\COIS4470_A1Q1.exe (process 4904) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Observations:

- As we can see, Case 2 has a better results than Case 1. This points towards the assumption that usually only 1 server fails but since there were no back up servers, therefore, the percentage of days when company could do business on time were only ~81% and gets as low as ~67% with increasing failure probability
- Having a back up server not only improves the percentage of days when company could do business on time, but also, the ratio of the decrease in this percentage with rising failure probability also decreases ○ In Case 1 our results decrease from ~81% to ~67% ○ While, in Case 2, our results decrease from ~98% to only ~94%

Question 2 : Newspaper Seller's Problem

Simulation Results for the Newspaper Seller's Problem

The program will simulate and give us the total profit earned for different cases. Optimal number would be the Number of Newspapers that one should buy to receive the most

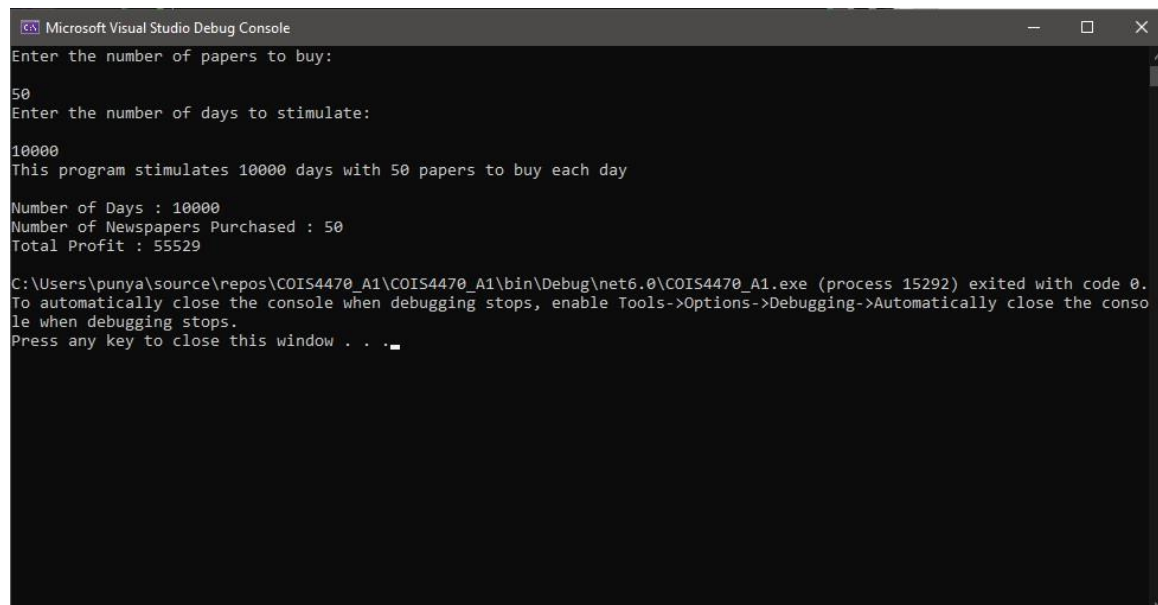
Number of Newspapers	40	50	60	70	80	90	100	Optimal Number
1000 days	2682	5586	7774	9054	9599	9556	9121	80
10000 days	27140	55529	78432	90932	96097	95087	90718	80
100000 days	271712	555400	783960	908623	963169	949466	909784	80

Observations:

- Its interesting to note how Number of newspapers = 80 is always yielding the highest profit. The simulation was ran quite a number of times to make sure that the results are not “biased” but every time ‘80’ gave the highest profit.
- Another observation made is that as the number of days increase by magnitude of 10, so does the profit. For every case and scenario the profit increase was directly proportional to the number of days increase.
- As the number of newspapers increase for a particular number of days, the percentage of the growth also decreases in a high steep. For example:
 - Number of Days = 1000 ○ Growth of profit from newspapers bought = 40 to 50 is ~108%
 - Growth of profit from newspapers bought = 50 to 60 is ~39%
 - Growth of profit from newspapers bought = 60 to 70 is ~16%
 - Growth of profit from newspapers bought = 70 to 80 is ~6%
 - Decrease in profit from newspapers bought = 80 to 90 is -0.44%
 - Decrease in profit from newspapers bought = 90 to 100 is -4%As we see here, the graph rises until it reaches our optimal number of newspapers bought and then decreases.

Not all the screenshots were attached, because there are a LOT of cases. But, attaching ONE screenshot of the output:

Number of paper = 50 Number of days = 10000



```
Microsoft Visual Studio Debug Console
Enter the number of papers to buy:
50
Enter the number of days to stimulate:
10000
This program stimulates 10000 days with 50 papers to buy each day
Number of Days : 10000
Number of Newspapers Purchased : 50
Total Profit : 55529
C:\Users\punya\source\repos\COIS4470_A1\COIS4470_A1\bin\Debug\net6.0\COIS4470_A1.exe (process 15292) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

(Question 3 starts on the Next Page to make the document look clean)

Question 3 : A Single Server Queue Simulation

This question was written in Python as there were a lot of factors involved and I believed Python would be a friendly language to write the program in.

Part a : Modify the problem to calculate “maximum delay”, “proportion of jobs delayed” and “number of jobs in system at specified time” Following changes were made to the program:

- **Maximum Delay:** A new variable was added in the *class sum* called `maxDelay=0.0`. This variable uses the Python function “max” to calculate the maximum of the values between the current delay and the already stored `maxDelay` value. The value is initialized to 0.0 since the initial delay is 0. Then every loop iteration we check if there is a delay, and if there is, then we check if this current delay is greater than the `maxDelay` value, if yes then `maxDelay` value is updated, otherwise it stays the same.
- **Proportions of jobs delayed :** To calculate this, a variable was created in the main program called the “`delayedJobs`” and every time there is a delay this variable is incremented by 1
- **Number of jobs at the specified time:** A variable that stores the specified time value was created and counter was created to keep track of number of jobs in system. In the loop, it is checked if the current departure time is greater than this specified time value and if current arrival time is less than specified time. If it is then we add 1 to the counter. Same thing is repeated for each specified time.

Part b : Small number of input

a_i (arrival time)	s_i (service time)
15	43
47	36
71	34
111	30
123	38
152	40
166	31
226	29
310	36
320	30

This list was added as another .dat file called “ssq2.dat” file. The program reads the input and gives us the following results.

```

=== Starting Single Server Simulation for FIFO ===

for 10 jobs
    average interarrival time .. = 32.00
    average service time ..... = 34.70
    proportions of delayed jobs = 80.00%
    maximum delay ..... = 70.00
    average delay ..... = 26.70
    average wait ..... = 61.40

    Number of jobs in the system at time 300 is 0
>>>

```

Part c : ssq1.dat file as input

Now running the program for the ssq1.dat file as input

```

=== Starting Single Server Simulation for FIFO ===

for 1000 jobs
    average interarrival time .. = 9.87
    average service time ..... = 7.12
    proportions of delayed jobs = 72.30%
    maximum delay ..... = 118.76
    average delay ..... = 18.59
    average wait ..... = 25.72

    Number of jobs in the system at time 2000 is 10

    Number of jobs in the system at time 6000 is 4
>>>

```

Observations:

- A very clear and obvious observation is that as the number of jobs increase from 10 to 1000, the average of everything decreased drastically – as it should have, otherwise something is wrong with the program :)
- An interesting thing to note is how the maximum delay increases by a very high magnitude. This may point to the fact that the arrival times of the jobs might be very close to each other while service times may be higher
- The simulation gives statistics (numbers) to support a clear and expected conclusion, that if a high traffic (high number of jobs) are expected, then a single server would not be efficient
- It will be interesting to try and write a simulation to choose an efficient number of servers given a certain number of expected jobs