COIS 4470H Assignment 3

Question 1: GPSS 01

Language: GPSS

Interarrival Time = RVEXPO(1,30) #in seconds

Service Time = 40_-4 seconds

a. <=0.58 go to Morning Paper Guy. Everyone else goes to Wall Street Journal Guy (even who are buying both papers)

A3Q1a.gps | A3Q1A.LIS

A3Q1a.gps

```
            SIMULATE
            REALLOCATE      COM,20000

            GENERATE    RVEXPO(1,30)
            TRANSFER    .58,WALL,MORN

MORN        QUEUE       LINE1
            SEIZE       VENDOR1
            DEPART      LINE1

            ADVANCE     40,4
            RELEASE     VENDOR1
            TERMINATE

WALL        QUEUE       LINE2
            SEIZE       VENDOR2
            DEPART      LINE2

            ADVANCE     40,4
            RELEASE     VENDOR2
            TERMINATE

            GENERATE    7200
            TERMINATE   1

            START       1
            END
```

```
     |   |  --AVG-UTIL-DURING--
FACILITY  TOTAL  AVAIL  UNAVL     ENTRIES    AVERAGE   CURRENT  PERCENT  SEIZING  PREEMPTING
     |   |  TIME   TIME   TIME               TIME/XACT   STATUS   AVAIL     XACT      XACT
 VENDOR1  0.683                          124     39.642    AVAIL              215
 VENDOR2  0.498                           91     39.435    AVAIL              218


   QUEUE      MAXIMUM      AVERAGE      TOTAL       ZERO     PERCENT     AVERAGE      $AVERAGE     QTABLE     CURRENT
     |   |    CONTENTS     CONTENTS     ENTRIES     ENTRIES    ZEROS     TIME/UNIT    TIME/UNIT    NUMBER     CONTENTS
   LINE1            4        0.660        127         29        22.8      37.441       48.521                      3
   LINE2            4        0.219         93         49        52.7      16.972       35.873                      2


  RANDOM     ANTITHETIC     INITIAL      CURRENT      SAMPLE   CHI-SQUARE
  STREAM       VARIATES     POSITION     POSITION      COUNT   UNIFORMITY
     |   1          OFF      100000       100656        656       0.72
```

Vendor1 = Morning Paper Guy

Vendor2 = Wall Street Journal Guy

Vendor	Number of People/ Contents	Time
Morning Paper Guy	124	Total: 0.683 Average: 39.642
Wall Street Journal	91	Total: 0.498 Average: 39.435

Queue	Number of People/ Contents	Time
Line1 (Morning)	Total: 127 Max: 4 Average: 0.660	Average Time: 37.441
Line2 (Wall Street)	Total: 93 Max: 4 Average: 0.219	Average Time: 16.972

b. How to make the system more efficient?

As we can see, the average time in the Queue for the Morning Paper Vendor, is very high as compared to Wall Street Journal Guy. Since the probability of people buying morning paper is 58, which is almost 60% - that is a high probability.

We know that Wall Street Journal Vendor has both Morning Paper and Wall Street Journals. Therefore, To optimize the system, we can implement the method where the incoming customers for morning papers are DISTRIBUTED among the two vendors. That is, the customer who wants to buy the morning paper will be sent to the Wall Street Journal Vendor if the Wall Street Journal has the shorter queue

A3Q1a.gps | A3Q1b.gps | A3Q1B.LIS

A3Q1b.gps

```
        REAL &U15
        LET &U15=0

        REAL &U30
        LET &U30=0

        GENERATE   RVEXPO(1,30)
        TRANSFER   .58,WALL,MORN

MORN    TEST LE &U15,&U30,WALL
        QUEUE      LINE1
        BLET &U15=&U15+1

        SEIZE      VENDOR1
        DEPART     LINE1
        BLET &U15=&U15-1

        ADVANCE    40,4
        RELEASE    VENDOR1

        TERMINATE

WALL    QUEUE      LINE2
        BLET &U30=&U30+1

        SEIZE      VENDOR2
        DEPART     LINE2
        BLET &U30=&U30-1

        ADVANCE    40,4
        RELEASE    VENDOR2
        TERMINATE

        GENERATE   7200
        TERMINATE  1

        START      1
        END
```

```
          --AVG-UTIL-DURING--
FACILITY  TOTAL  AVAIL  UNAVL    ENTRIES    AVERAGE   CURRENT  PERCENT  SEIZING  PREEMPTING
           TIME   TIME   TIME              TIME/XACT   STATUS   AVAIL     XACT      XACT
 VENDOR1  0.602                       109     39.764    AVAIL              221
 VENDOR2  0.611                       111     39.660    AVAIL              220


  QUEUE      MAXIMUM     AVERAGE      TOTAL      ZERO     PERCENT    AVERAGE     $AVERAGE    QTABLE     CURRENT
             CONTENTS    CONTENTS     ENTRIES    ENTRIES   ZEROS     TIME/UNIT   TIME/UNIT   NUMBER     CONTENTS
  LINE1            2       0.215        109        47       43.1       14.230      25.018                     0
  LINE2            3       0.404        111        38       34.2       26.195      39.831                     0


  RANDOM    ANTITHETIC    INITIAL     CURRENT     SAMPLE   CHI-SQUARE
  STREAM      VARIATES    POSITION    POSITION     COUNT   UNIFORMITY
       1           OFF     100000      100661        661      0.77
```

As we see, the average time for the Morning Paper line decreased drastically.

One more thing that we could do, the very obvious thing is add another vendor. This way we have 3 servers now so that would decrease the total time. But having one more vendor depends on 'budget' realistically speaking. So, the above solution would be better option to make it efficient.

Question 2: GPSS 02

Performance Measure	**LoadSplit**	**TurnTaker**	**ShortQ**
Server Utilization – Server 1	Total = 0.770 Avg time/Xact = 100.000	Total = 0.822 Avg time/Xact = 100.00	Total = 0.934 Avg time/Xact = 98.765
Server Utilization – Server 2	Total = 0.910 Avg time/Xact = 98.801	Total = 0.859 Avg time/Xact = 98.472	Total = 0.857 Avg time/Xact = 100.000
Number of request served - Server 1	46	49	52
Number of request served - Server 2	54	51	48
Average number in queue – queue for server 1	2.010	2.482	0.990
Average number in queue – queue for server 2	1.936	2.843	0.613
Mean delay time - queue for server 1	239.981	269.082	102.747
Mean delay time - queue for server 2	195.947	325.884	71.509
Mean Response Time – Server 1 (total waiting time for each request)	339.981	369.082	201.512
Mean Response Time – Server 2	294.748	424.356	171.509

The obvious recommendation would be the “ShortQ”. The server utilization depends on the shorter queue length, which means, there is not going to be an unnecessary crowd and load on one server just because of some ‘random number probability’.

Taking the statistics and performance of the simulation into consideration to support my recommendation –

- The server utilization for both the servers are ‘equal’ or similar. This means, that both servers are being used in a similar way and just one server does NOT have excess load
- The number of requests served by both the Web servers are again, similar – (51 and 49). Since we have assumed that all the requests take exactly 100.00 time for processing therefore, then this similarity indicates that both servers’ utilization is very close
- Average number in queue and the mean delay time as you can see makes a BIG difference. The requests are being serviced at a much faster rate as compared to other servers
- Overall, The mean response time is the lowest for ShortQ and hence the best approach

Load Split

A3Q2_L.gps | A3Q2LO | A3Q2_T.gps

A3Q2LO

```
1    Server 1 is   46
2    Server 2 is  54
3
```

FACILITY	--AVG-UTIL-DURING-- TOTAL TIME	AVAIL TIME	UNAVL TIME	ENTRIES	AVERAGE TIME/XACT	CURRENT STATUS	PERCENT AVAIL	SEIZING XACT	PREEMPTING XACT
SERVER1	0.770			46	100.000	AVAIL			
SERVER2	0.910			55	98.801	AVAIL		104	

QUEUE	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	PERCENT ZEROS	AVERAGE TIME/UNIT	$AVERAGE TIME/UNIT	QTABLE NUMBER	CURRENT CONTENTS
LINE1	8	2.010	50	10	20.0	239.981	299.976		4
LINE2	6	1.936	59	7	11.9	195.947	222.324		4

RANDOM STREAM	ANTITHETIC VARIATES	INITIAL POSITION	CURRENT POSITION	SAMPLE COUNT	CHI-SQUARE UNIFORMITY
1	OFF	100000	100219	219	0.76

Turn Taker

```
Server 1 is    49
Server 2 is   51
```

```
              --AVG-UTIL-DURING--
FACILITY  TOTAL  AVAIL  UNAVL      ENTRIES    AVERAGE   CURRENT  PERCENT  SEIZING  PREEMPTING
          TIME   TIME   TIME                  TIME/XACT STATUS   AVAIL    XACT     XACT
 SERVER1  0.822                          49   100.000    AVAIL
 SERVER2  0.859                          52    98.473    AVAIL              93


   QUEUE       MAXIMUM      AVERAGE      TOTAL      ZERO     PERCENT     AVERAGE     $AVERAGE      QTABLE     CURRENT
               CONTENTS     CONTENTS     ENTRIES    ENTRIES  ZEROS       TIME/UNIT   TIME/UNIT     NUMBER     CONTENTS
   LINE1             9        2.482          55         5       9.1      269.082     295.990                         6
   LINE2            10        2.843          52         6      11.5      325.884     368.391                         0


  RANDOM     ANTITHETIC      INITIAL     CURRENT      SAMPLE   CHI-SQUARE
  STREAM       VARIATES     POSITION    POSITION       COUNT   UNIFORMITY
       1            OFF       100000      100108         108     0.39
```

ShortQ

```
A3Q2_T.LIS    A3Q2SO

A3Q2SO
1        Server 1 is    52
2        Server 2 is   48
3
```

```
              --AVG-UTIL-DURING--
FACILITY  TOTAL  AVAIL  UNAVL      ENTRIES    AVERAGE   CURRENT  PERCENT  SEIZING  PREEMPTING
          TIME   TIME   TIME                  TIME/XACT STATUS   AVAIL    XACT     XACT
 SERVER1  0.934                          53    98.765    AVAIL             101
 SERVER2  0.857                          48   100.000    AVAIL


   QUEUE       MAXIMUM      AVERAGE      TOTAL      ZERO     PERCENT     AVERAGE     $AVERAGE      QTABLE     CURRENT
               CONTENTS     CONTENTS     ENTRIES    ENTRIES  ZEROS       TIME/UNIT   TIME/UNIT     NUMBER     CONTENTS
   LINE1             4        0.990          54         8      14.8      102.747     120.616                         1
   LINE2             3        0.613          48        15      31.2       71.509     104.013                         0


  RANDOM     ANTITHETIC      INITIAL     CURRENT      SAMPLE   CHI-SQUARE
  STREAM       VARIATES     POSITION    POSITION       COUNT   UNIFORMITY
       1            OFF       100000      100103         103     0.37
```

Question 3: Random Number Generator

Lehmer random-number generator

x_{i+1} = g(x) = ax_i mod 251 *i=0,1,2,3…….*

m = 251

a. Using Theorem: "If m is prime and p1, p2, . . . , pr are the (unique) prime factors of m − 1"

m-1 = 250 = 2 * 5^3

number of full-period multipliers = (p1-1)(p2-1)(m-1)/(p1*p2)

= (2-1)(5-1)(251-1)/(2*5)

= (250*4)/10 = 100

There are **100** full – period multipliers

b. Program:

```
PS C:\Users\punya\Documents\Priyam\2023 Winter\COIS 4470H\Assignments\A3>
PS C:\Users\punya\Documents\Priyam\2023 Winter\COIS 4470H\Assignments\A3> python3.8 A3Q3_FPM.py > A3_all_multipliers.txt
PS C:\Users\punya\Documents\Priyam\2023 Winter\COIS 4470H\Assignments\A3>
```

```
A3_all_multipliers.txt - Notepad
File Edit Format View Help
m =  251
Number of multipliers:  100

6
11
14
18
19
24
26
29
30
33
34
37
42
43
44
46
53
54
55
```

Please check entire output in attached file A3_all_multipliers.txt

c. Output File: A3_random_numbers.txt

Generating random numbers in range (0,1)

x0 = 3

```
 Please enter seed 'x0' value: 3
Generating random numbers uniformly from (0,1):

0.07171314741035857
0.4302788844621514
0.5816733067729084
0.4900398406374502
0.9402390438247012
0.6414342629482072
0.848605577689243
0.09163346613545817
0.549800796812749
0.29880478087649404
0.7928286852589641
0.7569721115537849
0.5418326693227091
0.250996015936255
0.5059760956175299
0.035856573705179286
0.2151394422310757
0.2908366533864542
0.7450199203187251
0.4701195219123506
0.8207171314741036
0.9243027888446215
0.545816733067729
0.2749003984063745
0.649402390438247
0.896414342629482
0.3784860557768924
0.27091633466135456
0.6254980079681275
0.7529880478087649
0.5179282868525896
0.10756972111553785
0.6454183266932271
0.8725099601593626
0.2350597609561753
0.4103585657370518
0.46215139442231074
0.7729083665338645
0.6374501992031872
0.8247011952191236
0.9482071713147411
0.6892430278884463
0.13545816733067728
0.8127490039840638
0.8764940239043825
0.2589641434262948
0.5537848605577689
0.32270916334661354
```

d. Output File: A3_100RN

Making a change to code so 100 random variates are generated

```python
def generate_random_numbers(seed: int, a: int, m: int, n: int) -> list:
    numbers = list()
    x = seed

    for i in range(n):
        x = (x*a) % m
        numbers.append(x/m)
        if x == seed: break

    return numbers



if __name__ == '__main__':
    x0 = int(input("\n Please enter seed 'x0' value: "))
    n = 100
    m = 251
    a = get_period_multiplier(m)

    random_numbers = generate_random_numbers(x0,a,m,100)

    print("Generating random numbers uniformly from (0,1): \n")
    for num in random_numbers:
        print(num)
```

```
 Please enter seed 'x0' value: 3
Generating random numbers uniformly from (0,1):

0.07171314741035857
0.4302788844621514
0.5816733067729084
0.4900398406374502
0.9402390438247012
0.6414342629482072
0.848605577689243
0.09163346613545817
0.549800796812749
0.29880478087649404
0.7928286852589641
0.7569721115537849
0.5418326693227091
0.250996015936255
0.5059760956175299
0.035856573705179286
0.2151394422310757
0.2908366533864542
0.7450199203187251
0.4701195219123506
0.8207171314741036
0.9243027888446215
0.545816733067729
0.2749003984063745
0.649402390438247
0.896414342629482
0.3784860557768924
0.27091633466135456
0.6254980079681275
0.7529880478087649
0.5179282868525896
```

e. Chi squared test

Chi squared statistic value generated = 12.2

For alpha = 0.01 and s = 10, (degree of freedom = 9) chi critical value = 21.7

12.2 is less than 21.7, therefore, TRUE all values generated are uniform

I anyways implemented the calculation of this in the program itself. As you see, it says “True” for being uniformly generated

```
 Please enter seed 'x0' value: 3

 chi squared statistic value :  12.2
Generating random numbers uniformly from (0,1)

USing chi-squared test, Are the random numbers uniformly distributes:  True
0.07171314741035857
0.4302788844621514
0.5816733067729084
0.4900398406374502
0.9402390438247012
0.6414342629482072
0.848605577689243
0.09163346613545817
0.549800796812749
0.29880478087649404
0.7928286852589641
0.7569721115537849
0.5418326693227091
0.250996015936255
0.5059760956175299
0.035856573705179286
0.2151394422310757
0.2908366533864542
0.7450199203187251
0.4701195219123506
0.8207171314741036
0.9243027888446215
0.545816733067729
0.2749003984063745
0.649402390438247
0.896414342629482
0.3784860557768924
0.27091633466135456
0.6254980079681275
0.7529880478087649
0.5179282868525896
0.10756972111553785
0.6454183266932271
0.8725099601593626
0.2350597609561753
0.4103585657370518
0.46215139442231074
0.7729083665338645
```

f. GAP Test (0.2, 0.5)

Output File: A3_GAP_Out.txt

Random numbers are independent :)

```
 Please enter seed 'x0' value: 3

 Random Numbers in range:  32

 Number of gaps :  31

 Printing Gap length and fo:

{1: 8, 5: 1, 3: 3, 2: 6, 0: 9, 6: 1, 8: 2, 11: 1}

 Printing fe values:

[6.51, 1.5630509999999993, 3.1898999999999993, 4.556999999999999, 9.299999999999999, 1.0941356999999996, 0.5361264929999997, 0.18389138709899988]

 X2 chi :  8.648806634645176

 Table value:  52.19139483319193

 Random Numbers are independent!
Generating random numbers uniformly from (0,1)

0.07
0.43
0.58
0.49
0.94
0.64
0.85
0.09
0.55
0.3
0.79
0.76
0.54
0.25
0.51
0.04
0.22
0.29
0.75
0.47
0.82
0.92
0.55
0.27
0.65
0.9
0.38
```