CS-342

End-Semester

Name: P. V. Sriram

Roll No.: 1801CS37

Q1a)

To Compile: gcc Q1a.c -o Q1a

To Run: ./Q1a

Input:

a) Enter the resources required by each process (c): 3
   Enter the total number of resources available (r): 4
b) Enter the resources required by each process (c): 6
   Enter the total number of resources available (r): 3
c) Enter the resources required by each process (c): 3
   Enter the total number of resources available (r): 6

Output:

```
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ gcc Q1a.c -o Q1a
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ ./Q1a
Enter the resources required by each process (c): 3
Enter the total number of resources available (r): 4
Maximum number of resources to ensure deadlock free scenario(n): 1
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ ./Q1a
Enter the resources required by each process (c): 6
Enter the total number of resources available (r): 3
Maximum number of resources to ensure deadlock free scenario(n): 0
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ ./Q1a
Enter the resources required by each process (c): 3
Enter the total number of resources available (r): 6
Maximum number of resources to ensure deadlock free scenario(n): 2
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$
```

Note: The above answers are following the formula (r - 1) / (c - 1). Except for when c = 1, in which case answer is Infinite

Q1b)

To Compile: gcc Q1a.c -o Q1a

To Run: ./Q1a

Input:

a) Enter the number of processes: 3
   Enter the requirements of processes: 6 7 9
b) Enter the number of processes: 3
   Enter the requirements of processes: 1 4 5
c) Enter the number of processes: 4
   Enter the requirements of processes: 35 45 54 20

Output:

```
sriram@sriram: ~/CS-342 OS Lab/End-Sem
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ gcc Q1b.c -o Q1b
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ ./Q1b
Enter the number of processes: 3
Enter the requirements of processes: 6 7 9
Maximum resources for non guaranteed deadlock free operation: 19
Minimum resources for guaranteed deadlock free operation: 20
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ ./Q1b
Enter the number of processes: 3
Enter the requirements of processes: 1 4 5
Maximum resources for non guaranteed deadlock free operation: 7
Minimum resources for guaranteed deadlock free operation: 8
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ ./Q1b
Enter the number of processes: 4
Enter the requirements of processes: 35 45 54 20
Maximum resources for non guaranteed deadlock free operation: 150
Minimum resources for guaranteed deadlock free operation: 151
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$
```

Note: The above answers are result of the formulae

Max for non-guarantee deadlock free: Sum(Ci) – n

Min for guarantee deadlock free: Sum(Ci) – n + 1

Q2)

LAMA

- The algorithm I have implemented is called LAMA. It is a variation of LFU.

- While LFU uses the past data to figure out least frequently occurring pages. LAMA uses the future data to rate the pages.
- From the reference array, at one point in time, if there is a page miss, we look at the array elements right side to it and replace the least frequently occurring page in the RHS.
- Following data represents the algorithms performance

To Compile: g++ Q2.cpp -o Q2

To Run: ./Q2

Input

a) Enter Number of Trials: 2
Enter Number of Frames: 3
Length of Page Sequence: 10
4 7 6 1 7 6 1 2 7 2

b) Enter Number of Frames: 4
Length of Page Sequence: 16
1 2 3 4 2 7 5 1 1 6 4 7 2 1 2 5

Output

```
^[[A(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ g++ Q2.cpp -o Q2
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ ./Q2
Enter Number of Trials: 2
Enter Number of Frames: 3
Length of Page Sequence: 10
4 7 6 1 7 6 1 2 7 2
LAMA:
4 7 6 1 7 6 1 2 7 2
#Page Faults: 5
#Successful Hits: 5
#Success Ratio: 0.500
****************************************
FIFO:
4 7 6 1 7 6 1 2 7 2
#Page Faults: 6
#Successful Hits: 4
#Success Ratio: 0.400
****************************************
LRU:
4 7 6 1 7 6 1 2 7 2
#Page Faults: 6
#Successful Hits: 4
#Success Ratio: 0.400
****************************************
Enter Number of Frames: 4
Length of Page Sequence: 16
1 2 3 4 2 7 5 1 1 6 4 7 2 1 2 5
LAMA:
4 7 6 1 7 6 1 2 7 2 1 2 3 4 2 7 5 1 1 6 4 7 2 1 2 5
#Page Faults: 11
#Successful Hits: 15
#Success Ratio: 0.577
****************************************
FIFO:
4 7 6 1 7 6 1 2 7 2 1 2 3 4 2 7 5 1 1 6 4 7 2 1 2 5
#Page Faults: 16
#Successful Hits: 10
#Success Ratio: 0.385
****************************************
LRU:
4 7 6 1 7 6 1 2 7 2 1 2 3 4 2 7 5 1 1 6 4 7 2 1 2 5
#Page Faults: 16
#Successful Hits: 10
#Success Ratio: 0.385
****************************************
Page Faults List for LAMA Algorithm: 5 11
Page Faults List for FIFO Algorithm: 6 16
Page Faults List for LRU Algorithm: 6 16
Execution times List for LAMA Algorithm: 0.000235 0.000139
Execution times List for FIFO Algorithm: 9.5e-05 2.3e-05
Execution times List for LRU Algorithm: 0.000136 3.6e-05
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$
```

Note:

- We can see above that LAMA Algorithm has 5 page faults in first case as compared to 6 in FIFO and LRU.
- And also that LAMA Algorithm has 11 page faults in second case as compared to 16 in FIFO and LRU.
- A significant Improvement in performance over the base line algorithms is observed in LAMA

- Although, Lama also takes more time to execute than other two algorithm as indicated by the data above

Q3)

To Compile: g++ Q3.cpp -o Q3

To Run: ./Q3

Input:

a) Enter Number of Processes: 7
Enter Time Quantum 1: 5
Enter Time Quantum 2: 10
Enter List of Processes (Arrival Time, Burst Time):
A 0 18
B 0 12
C 0 7
D 0 11
E 0 28
F 7 18
G 16 12
Enter Time Limit:
53

b) Enter Number of Processes: 5
Enter Time Quantum 1: 44
Enter Time Quantum 2: 8
Enter List of Processes (Arrival Time, Burst Time):
A 0 23
B 1 12
C 0 13
D 5 7
E 17 20
Enter Time Limit:
20

Output:

```
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ g++ Q3.cpp -o Q3
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ ./Q3
Enter Number of Processes: 7
Enter Time Quantum 1: 5
Enter Time Quantum 2: 10
Enter List of Processes (Arrival Time, Burst Time):
A 0 18
B 0 12
C 0 7
D 0 11
E 0 28
F 7 18
G 16 12
Enter Time Limit:
53
Job ID  Arrival Time    Execution Time  Start Time      End Time        Current  Status
A       0               18              0               43              C
B       0               12              0               24              C
C       0               7               5               22              C
D       0               11              10              30              C
E       0               28              10              -1              E
F       7               18              20              51              C
G       16              12              25              52              C
```

```
(base) sriram@sriram:~/CS-342 OS Lab/End-Sem$ ./Q3
Enter Number of Processes: 5
Enter Time Quantum 1: 4
Enter Time Quantum 2: 8
Enter List of Processes (Arrival Time, Burst Time):
A 0 23
B 1 12
C 0 13
D 5 7
E 17 20
Enter Time Limit:
20
Job ID  Arrival Time    Execution Time  Start Time      End Time        Current  Status
A       0               23              0               -1              W
C       0               13              0               20              C
B       1               12              4               -1              W
D       5               7               8               15              C
E       17              20              -1              -1              
```

Q4)

To Compile: g++ Q4.cpp -o Q4

To Run: ./Q4

Input

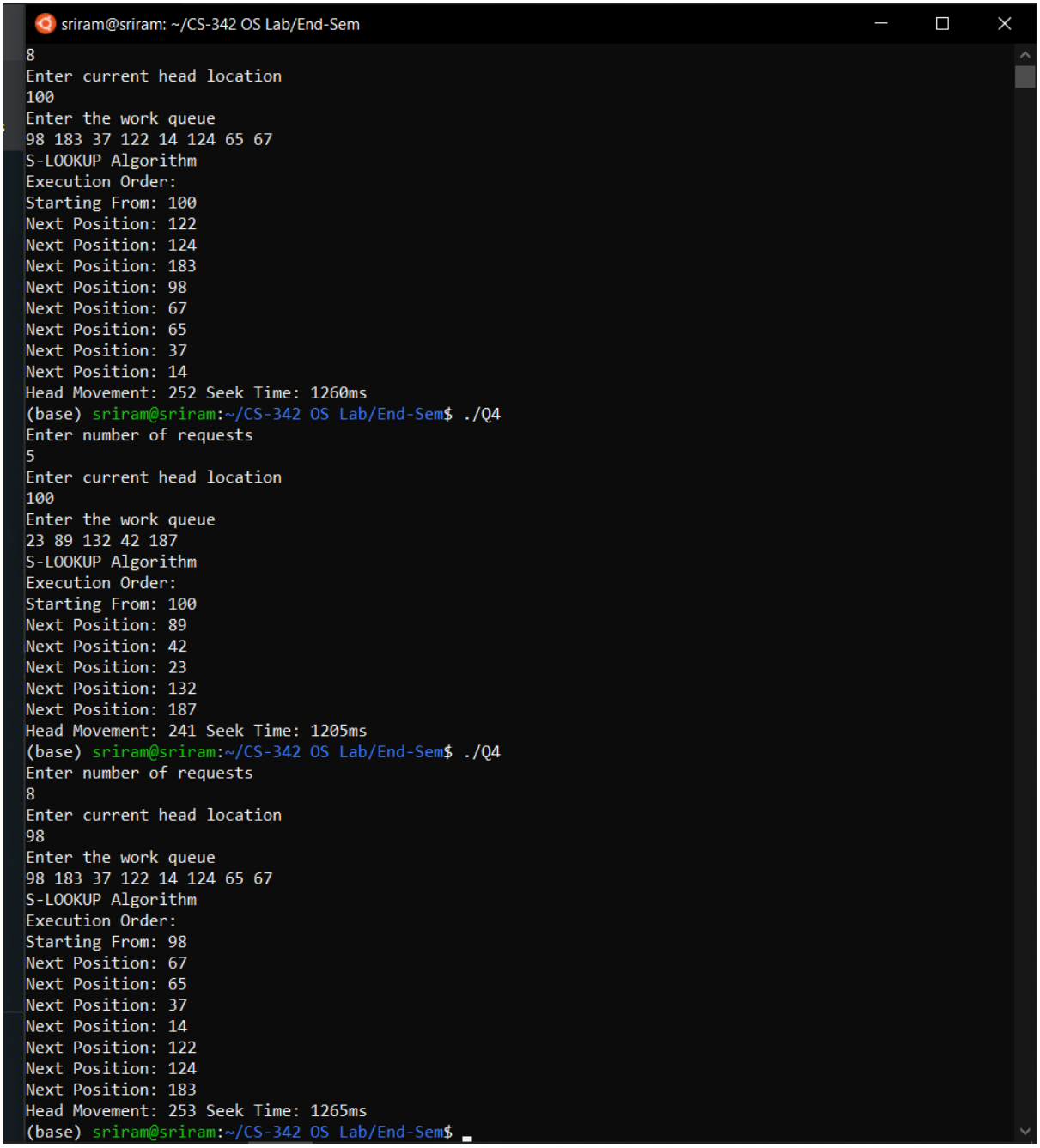
a) Enter number of requests
   8
   Enter current head location
   100

Enter the work queue
98 183 37 122 14 124 65 67

b) Enter number of requests
5
Enter current head location
100
Enter the work queue
23 89 132 42 187

c) Enter number of requests
8
Enter current head location
98
Enter the work queue
98 183 37 122 14 124 65 67

Output



We get Head movements of 252, 241, 253 from the three cases respectively.