CS 558: Computer Systems Lab (2020)

Assignment Set III

- **Assignments will be evaluated by the TAs.**
- **You should submit report (for answering non-code related questions, if any), complete source codes and executable files (whichever applicable).**
- **All codes must be properly documented and good code writing practice should be followed (carry marks).**
- **Copying is strictly prohibited. Any case of copying will automatically result in F grade for the whole course, irrespective of your performance in the other parts of the lab.**
- **Submission deadline: March 22, 2020.**
- **Marks distribution: 35, 30, 35.**

1. CPU Management – *Implementation of CPU Scheduling Algorithms*: (35 Marks)

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to execute. The selection process is carried out by the short-term scheduler (or CPU scheduler). The scheduler selects one process among the processes in memory that are ready to be executed, and allocates the CPU to the same. There are several algorithms for CPU scheduling. Some of the algorithms are preemptive, whereas some of them are non-preemptive. In case of non-preemptive scheduling, once a process is scheduled, it holds the CPU until it completes its execution. On the other hand, in preemptive scheduling, the process may have to leave the CPU before the completion of execution because of the higher priority of another process.

In this assignment, you are required to implement the Shortest Job First (SJF) CPU scheduling algorithm, both preemptive and non-preemptive versions using *C programming*. Your program should take the ready queue of processes, along with the arrival time and CPU burst time of each process as input, and display the following as outputs.

(i) The 'Gantt Chart' for the processes, both for the preemptive and non-preemptive versions of the algorithm;

(ii) The average waiting time and average turnaround time; both for the preemptive and non-preemptive versions of the algorithm; and

(iii) Percentage of performance improvement of preemptive-scheduling over the non-preemptive scheduling.

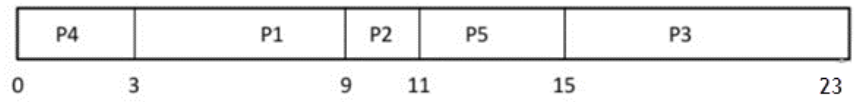
Sample input and output:

Example Input File (test case may be different from this file during evaluation):

```
Process Queue  Burst time     Arrival time
P1                     6              2
P2                     2              5
P3                     8              1
P4                     3              0
P5                     4              4
```

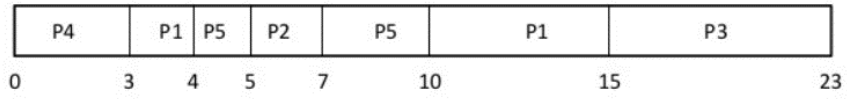
Example Output:

```
Gantt chart for non-preemptive SJF
```




```
Average Waiting Time= (1+4+14+0+7)/5 = 26/5 = 5.2
Average Turnaround Time = (7+6+22+3+11)/5 = 49/5 = 9.8
```

```
Gantt chart for preemptive SJF
```




```
Average Waiting Time= (7+0+14+0+2)/5 = 23/5 =4.6
Average Turnaround Time = (13+2+22+3+6)/5 = 46/5 = 9.2

Improvement of performance of preemptive over non-preemptive SJF scheduling:
In terms of average waiting time = 11.54%
In terms of turnaround time = 06.52%
```

2. Memory Management – *Implementation of Page Replacement Algorithms*: (30 Marks)

Virtual pages are mapped to physical frames based on demands. This is required because the virtual memory is huge in general (conceptually unlimited), whereas the physical memory is limited. There are different schemes for the page replacement including the FCFS (First Come First Serve), LRU (Least Recently Used), and Optimal page replacement algorithm. In this assignment, you have to implement two algorithms namely, the LRU (using stack only) and Optimal, with a set of page reference sequence and the number of available frames in the physical memory as inputs. As output, you have to show the number of page fault and page fault ratios for the two algorithms for the given page sequence and specified number of frames.

Sample input and output:

Example Input (test case may be different from this file during evaluation):

```
Page reference sequence: 1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4,
3
Number of frames: 3
```

Example Output:

```
Page fault ratio in case of optimal algorithm: 9/20 = 0.45
Page fault ratio in case of LRU algorithm: 11/20 = 0.55
```

Note: Showing the details of the page fault occurrences is encouraged (and carry marks).

```
Example:
Page reference stream:
1 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3
-----------------------------------------
1 1 1 1 1 1 1 1 6 6 6 6 6 6 6 6 6 2 2 2
  2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 4 4
    3 3 3 5 5 5 5 5 5 5 3 3 3 3 3 3 3 3
* * *     *     *       * *       * *
Optimal
Total 9 page faults
```

Page reference stream:

1 2 3 2 1 5 2 1 6 2 5 6 3 1 3 6 1 2 4 3

1	1	1	1	3	2	1	5	2	1	6	2	5	6	6	1	3	6	1	2
	2	2	3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4
		3	2	1	5	2	1	6	2	5	6	3	1	3	6	1	2	4	3
*	*	*			*			*		*		*	*				*	*	*

LRU

Total 11 page faults

3. Elevator Control System: (35 Marks)

In this question, you are required to emulate an Elevator control system for a skyscraper, called *Burj Khalifa*, with 163 floors (assuming the ground floor is marked as floor 1). Your program should take a file and the direction of Elevator (upward or downward) as input using command line argument. The Elevator can change the direction only after reaching the last floor in the current direction. The input file contains several rows. Each row in the input file has two entries namely, current Elevator floor position and the new floor requests received while the Elevator is at the current floor. In addition, the first entry of the first row in the file represents the initial floor position of the Elevator. The rest of the rows in the file contain floor number (as first entry) and the additional requests received after reaching that floor. It should be noted that the additional requests can have few redundant floor requests. Your program should perform the necessary validations checks. A sample file format is provided (see below) for your reference. Your program should calculate the total movement of the Elevator and the order in which the floor requested are serviced.

Input format: *.\a.out filename.txt upward*

Sample file format:

Floor Number	Floor Requests Received
55	35, 28, 82, 148
28	21, 63, 81
148	76, 35
63	159, 11, 81

Note: Floors where no additional requests were received are not included in the file.

Output of the Program:

```
Initial Elevator floor position: 55
Request Queue at Floor 55: 28 35 82 148

Reached Floor 82
Request Queue at Floor 82: 28 35 148

Reached Floor 148
Request Queue at Floor 148: 28 35 76

...
...
Total movement of the Elevator: 584
Order in which the floor requests are serviced:
55 - 82 - 148 - 76 - 35 - 28 - 21 - 63 - 81 - 159 - 11
```

Note:

(a) Your program is also required to print appropriate messages as and when required to demonstrate its execution sequence.

(b) During evaluation, the TA will provide different test cases (i.e., file with necessary information) as input. Your program should execute with the given custom inputs.