

# CS 4310 Operating Systems Project Simulating Job Scheduler and Performance Analysis

## ***Description:***

Simulating Job Scheduler of the Operating Systems by programming the following four scheduling algorithms that we covered in the class:

- a. First-Come-First-Serve (FCFS)
- b. Shortest-Job-First (SJF)
- c. Round-Robin with Time Slice = 2 (RR-2)
- d. Round-Robin with Time Slice = 5 (RR-5)

The project generates multiple testing cases with inputs of 5 jobs, 10 jobs, and 15 jobs, 20 cases each, for 60 files. The program will read process burst times from a file, job.txt, and execute the four algorithms above. A sample input file of five jobs is given as follows (burst time in ms). The program displays four algorithms job processes in a table format and calculates the average turnover time(ATT). It compares the mean of ATT for each job group.

[Begin of job.txt]

Job1

7

Job2

18

Job3

10

Job4

4

Job5

12

[End of job.txt]

Note: you can assume that

- (1) There are no more than 30 jobs in the input file (job.txt).
- (2) Processes arrive in the order they are read from the file for FCFS, RR-2 and RR-5.
- (3) All jobs arrive at time 0.
- (4) FCFS uses the order of the jobs, Job1, Job2, Job3, ...

**Part1**  
**Design & Testing (45 points)**

- a. Design the program by providing pseudocode or flowchart for each CPU scheduling algorithm.

<Insert answers here>

1. First Come First Serve (FCFS)

```
1 FCFS Pesudo code
2
3 /**
4  * The method simulate Fist Come First Serve(FCFS) scheduling algorithm
5  * @Param file jobList, the job list on the text file
6  * @Return double ataRslt, calculation result for Averatge Turnover Time
7  */
8 1. Read file
9 2. Make a data frame with integer 2D array // Method A
10 3. Create a job completion table
11 4. Make an empty array for Average Turnover Time
12 5. Record job completion // Method B
13 6. Display job completion table // Method C
14 7. Calculate Averatge Turnover Time // Method D
15 8. Dipaly the result.
16
17
18 //Method A
19 /**
20  * The method reads the text files and create 2D array
21  * Corresponding column 1:JobID, column2: rmnOfTime
22  * @Param file Object, jobList which shows the job ID and the remaining time
23  * @Return ArrayList<Integer><Integer> dataFrm, simulate Table with 2 columns
24  */
25 1. Create empty 2D array jobList
26 2. Set loop which itearates the data file until the end of files
27 3. Read current job id from the file object
28 4. Record the current job id on the data fram column 1
29 5. Read the job time
30 6. Record the job time on the data frae 2nd column
31
32 //Method B
33 /**
34  * The method accepts 2D integer arraylist data Frme
35  * It records the job process with anotehr 2D array
36  * Corresponding column 1:JobID, column 2: Start Time, column 3: End Time, column 4: complete job ID
37  * @Param ArayList<Integer><Integer> dataFrm, jobList which shows the job ID and the remaining time
38  * @Return ArrayList<Integer><Integer> jobRcd, job record which makes a work log until all the job is completed
39  */
40 1. Set start time <- 0, and end time <- 0
41 2. Set loop where an iterator itearates dataFrm and jobRcd from the first JobID to the last JobID (Top to Bottom)
42 3. Read JobID from dataFrm
43 4. Record JobID on JobRcd 1st column // JobID
44 5. Record startTime on JobRcd 2nd column // startTime
45 6. Read JobTime from dataFrm
46 7. Calculate endTime <- startTime + JobTime
47 8. Record endTime on JobRcd 3rd column // End Time
48 9. Record JobID on JobRcd 4th column as a completed task // Complete JobID
49 10. Upate startTime <- endTime
50 11. Move next row until the last JobID
51 12. Return jobRcd
52
```

```

52
53
54
55 //Method C
56 /**
57  * The method accepts 2D integer arraylist JobRcd
58  * It prints the job process with anotehr 2D array table format
59  * Corresponding column 1:JobID, column 2: Start Time, column 3: End Time, column 4: complete Job ID
60  * @Param ArrayList<Integer><Integer> jobRcd, job record which makes a work log until all the job is completed
61  * @Return void
62  */
63 1. Accept jobRcd
64 2. Create string headColumns
65 3. Create string jobCompletion
66 4. Set loop where an iterator iterates JobRcd from the 1st row to the last row
67 5. Display the 1st column // JobID
68 6. Display the 2nd column // Start Time
69 7. Display the 3rd coumn // End Time
70 8. Display the 4th column with the jobCompletion string // complete Job ID
71 9. Move to the nect row
72
73
74 //Method D
75 /**
76  * The method reads jobRcd and calculates the Average Turnover Time
77  * @Param ArrayList<Integer><Integer> jobRcd, job record which makes a work log
78  * @Return double aveTmovrTime, average turn over time.
79  */
80 1. Accept jobRcd
81 2. Set Su <- 0
82 2. Set loop where the iterator iterates the jobRcd's last column
83 3. Check the last column != 0
84 4. Read the same row column 3 // End Time
85 5. Add sum += endTime
86 6. Move to the next row
87 7. Devide the Sum by the number of row
88 8. Return the restult

```

## 2. Shortest Job First (SJF)

```
1 SJF Pesudo code
2
3 /**
4  * The method simulate Shortest Job First(SJF) scheduling algorithm
5  * @Param file jobList, the job list on the text file
6  * @Return double ataRslt, calculation result for Averatge Turnover Time
7  */
8 1. Read file
9 2. Make a data frame with integer 2D array // Method A
10 3. Sort the data frame decending order by the column 2 // Job Time
11 3. Create a job completion table
12 4. Make an empty array for Average Turnover Time
13 5. Record job completion // Method B
14 6. Display job completion table // Method C
15 7. Calculate Averatge Turnover Time // Method D
16 8. Dipaly the result.
17
18
19 //Method A
20 /**
21  * The method reads the text files and create 2D array
22  * Corresponding column 1:JobID, column2: rmnOfTime
23  * @Param file Object, jobList which shows the job ID and the remaining time
24  * @Return ArrayList<Integer><Integer> dataFrm, simulate Table with 2 columns
25  */
26 1. Create empty 2D array jobList
27 2. Set loop which iteartes the data file until the end of files
28 3. Read current job id from the file object
29 4. Record the current job id on the data fram column 1
30 5. Read the job time
31 6. Record the job time on the data frae 2nd column
32
33 //Method B
34 /**
35  * The method accepts 2D integer arraylist data Frme
36  * It records the job process with anotehr 2D array
37  * Corresponding column 1:JobID, column 2: Start Time, column 3: End Time, column 4: complete job ID
38  * @Param ArrayList<Integer><Integer> dataFrm, jobList which shows the job ID and the remaining time
39  * @Return ArrayList<Integer><Integer> jobRcrd, job record which makes a work log until all the job is completed
40  */
41 1. Set start time <- 0, and end time <- 0
42 2. Set loop where an iterator iteartes dataFrm and jobRcrd from the first JobID to the last JobId (Top to Bottom)
43 3. Read JobID from dataFrm
44 4. Record JobId on JobRcd 1st column // JobID
45 5. Record startTime on JobRcd 2nd column // startTime
46 6. Read JobTime from dataFrm
47 7. Calculate endTime <- startTime + JobTime
48 8. Record endTime on JobRcd 3rd column // End Time
49 9. Record JobID on JobRcd 4th column as a completed task // Complete JobID
50 10. Upate startTime <- endTime
51 11. Move next row until the last JobId
52 12. Return jobRcrd
53
```

```

55
56 //Method C
57 /**
58 * The method accepts 2D integer arraylist JobRcd
59 * It records the job process with anotehr 2D array table format
60 * Corresponding column 1:JobID, column 2: Start Time, column 3: End Time, column 4: complete Job ID
61 * @Param ArrayList<Integer><Integer> jobRcd, job record which makes a work log until all the job is completed
62 * @Return void
63 */
64 1. Accept jobRcd
65 2. Create string headColumns
66 3. Create string jobCompletion
67 4. Set loop where an iterator iterates JobRcd from the 1st row to the last row
68 5. Display the 1st column // JobID
69 6. Display the 2nd column // Start Time
70 7. Display the 3rd counn // End Time
71 8. Display the 4th column with the jobCompletion string // complete Job ID
72 9. Move to the nect row
73
74
75 //Method D
76 /**
77 * The method reads jobRcd and calculates the Average Turnover Time
78 * @Param ArrayList<Integer><Integer> jobRcd, job record which makes a work log
79 * @Return double aveTrnovrTime, average turn over time.
80 */
81 1. Accept jobRcd
82 2. Set Su <- 0
83 2. Set loop where the iterator iterates the jobRcd's last column
84 3. Check the last column != 0
85 4. Read the same row column 3 // End Time
86 5. Add sum += endTime
87 6. Move to the next row
88 7. Devide the Sum by the number of row
89 8. Return the restult

```

### 3. Round Robin Time Slice 2 (RR2)

```
1 RR2 Pseudo code
2
3 /**
4  * The method simulate Round Robin(RR) 2 scheduling algorithm
5  * @Param file jobList, the job list on the text file
6  * @Return double ataRslt, calculation result for Averatge Turnover Time
7  */
8 1. Read file
9 2. Make a data frame with integer 2D array // Method A
10 3. Sort the data frame decending order by the column 2 // Job Time
11 3. Create a job completion table
12 4. Make an empty array for Average Turnover Time
13 5. Record job completion // Method B
14 6. Display job completion table // Method C
15 7. Calculate Averatge Turnover Time // Method D
16 8. Dipaly the result.
17
18
19 //Method A
20 /**
21  * The method reads the text files and create 2D array
22  * Corresponding column 1:JobID, column2: rmnOfTime
23  * @Param file Object, jobList which shows the job ID and the remaining time
24  * @Return ArayList<Integer><Integer> dataFrm, simulate Table with 2 columns
25  */
26 1. Create empty 2D array jobList
27 2. Set loop which iteartes the data file until the end of files
28 3. Read current job id from the file object
29 4. Record the current job id on the data fram column 1
30 5. Read the job time
31 6. Record the job time on the data frae 2nd column
32
33 //Method B
34 /**
35  * The method accepts 2D integer arraylist data Frme
36  * It records the job process with anotehr 2D array
37  * Corresponding column 1:JobID, column 2: Start Time, column 3: End Time, column 4: complete job ID
38  * @Param ArayList<Integer><Integer> dataFrm, jobList which shows the job ID and the remaining time
39  * @Return ArayList<Integer><Integer> jobRcd, job record which makes a work log until all the job is completed
40  */
41 1. Set start time <- 0, and end time <- 0
42 2. Set RR <- 2
43 3. Set loop where an iterator iteartes dataFrm and jobRcd until the all the JobTime are 0
44 4. Calculate current row for jobRcd // i % the number of row in jobRcd
45 5. Read JobID from dataFrm
46 6. Record JobId on JobRcd 1st column // JobID
47 7. Record startTime on JobRcd 2nd column // startTime
48 8. Read JobTime from dataFrm
49 if JobTime > RR
50     Calculate endTime <- startTime + RR
51     Update JobTime <- JobTime - RR
52 else // JobTime =< RR
53     Calculate endTime <- startTime + JobTime
54     Update JobTime <- JobTime - JobTime // Make 0
55 9. Record endTime on JobRcd 3rd column // End Time
56 10. Record JobID on JobRcd 4th column as a completed task // Complete JobID
57 11. Upate startTime <- endTime
58 12. Move next row until the last JobId
59 13. Return jobRcd
60
```

```

63 //Method C
64 /**
65 * The method accepts 2D integer arraylist JobRcd
66 * It records the job process with anotehr 2D array table format
67 * Corresponding column 1:JobID, column 2: Start Time, column 3: End Time, column 4: complete Job ID
68 * @Param ArrayList<Integer><Integer> jobRcd, job record which makes a work log until all the job is completed
69 * @Return void
70 */
71 1. Accept jobRcd
72 2. Create string headColumns
73 3. Create string jobCompletion
74 4. Set loop where an iterator iterates JobRcd from the 1st row to the last row
75 5. Display the 1st column // JobID
76 6. Display the 2nd column // Start Time
77 7. Display the 3rd coumn // End Time
78 8. Display the 4th column with the jobCompletion string // complete Job ID
79 9. Move to the next row
80
81
82 //Method D
83 /**
84 * The method reads jobRcd and calculates the Average Turnover Time
85 * @Param ArrayList<Integer><Integer> jobRcd, job record which makes a work log
86 * @Return double aveTrnovrTime, average turn over time.
87 */
88 1. Accept jobRcd
89 2. Set Su <- 0
90 2. Set loop where the iterator iterates the jobRcd's last column
91 3. Check the last column != 0
92 4. Read the same row column 3 // End Time
93 5. Add sum += endTime
94 6. Move to the next row
95 7. Devide the Sum by the number of row
96 8. Return the restult

```

#### 4. Round Robin Time Slice 5

```

1  RR5 Pseudo code
2
3  /**
4   * The method simulate Round Robin(RR) 5 scheduling algorithm
5   * @Param file jobList, the job list on the text file
6   * @Return double ataRslt, calculation result for Averatge Turnover Time
7   */
8   1. Read file
9   2. Make a data frame with integer 2D array // Method A
10  3. Sort the data frame decending order by the column 2 // Job Time
11  3. Create a job completion table
12  4. Make an empty array for Average Turnover Time
13  5. Record job completion // Method B
14  6. Display job completion table // Method C
15  7. Calculate Averatge Turnover Time // Method D
16  8. Dipaly the result.
17
18
19  //Method A
20  /**
21   * The method reads the text files and create 2D array
22   * Corresponding column 1:JobID, column2: rmnOfTime
23   * @Param file Object, jobList which shows the job ID and the remaining time
24   * @Return ArrayList<Integer><Integer> dataFrm, simulate Table with 2 columns
25   */
26   1. Create empty 2D array jobList
27   2. Set loop which iteartes the data file until the end of files
28   3. Read current job id from the file object
29   4. Record the current job id on the data fram column 1
30   5. Read the job time
31   6. Record the job time on the data frae 2nd column
32
33  //Method B
34  /**
35   * The method accepts 2D integer arraylist data Frme
36   * It records the job process with anotehr 2D array
37   * Corresponding column 1:JobID, column 2: Start Time, column 3: End Time, column 4: complete job ID
38   * @Param ArrayList<Integer><Integer> dataFrm, jobList which shows the job ID and the remaining time
39   * @Return ArrayList<Integer><Integer> jobRcd, job record which makes a work log until all the job is completed
40   */
41   1. Set start time <- 0, and end time <- 0
42   2. Set RR <- 2
43   3. Set loop where an iterator iteartes dataFrm and jobRcd until the all the JobTime are 0
44   4. Calculate current row for jobRcd // i % the number of row in jobRcd
45   5. Read JobID from dataFrm
46   6. Record Jobld on JobRcd 1st column // JobID
47   7. Record startTime on JobRcd 2nd column // startTime
48   8. Read JobTime from dataFrm
49   if JobTime > RR
50       Calculate endTime <- startTime + RR
51       Update JobTime <- JobTime - RR
52   else // JobTime =< RR
53       Calculate endTime <- startTime + JobTime
54       Update JobTime <- JobTime - JobTime // Make 0
55   9. Record endTime on JobRcd 3rd column // End Time
56   10. Record JobID on JobRcd 4th column as a completed task // Complete JobID
57   11. Upate startTime <- endTime
58   12. Move next row until the last Jobld
59   13. Return jobRcd
60

```



```

63 //Method C
64 /**
65 * The method accepts 2D integer arraylist JobRcd
66 * It records the job process with anotehr 2D array table format
67 * Corresponding column 1:JobID, column 2: Start Time, column 3: End Time, column 4: complete Job ID
68 * @Param ArrayList<Integer><Integer> jobRcd, job record which makes a work log until all the job is completed
69 * @Return void
70 */
71 1. Accept jobRcd
72 2. Create string headColumns
73 3. Create string jobCompletion
74 4. Set loop where an iterator iterates JobRcd from the 1st row to the last row
75 5. Display the 1st column // JobID
76 6. Display the 2nd column // Start Time
77 7. Display the 3rd coumn // End Time
78 8. Display the 4th column with the jobCompletion string // complete Job ID
79 9. Move to the next row
80
81
82 //Method D
83 /**
84 * The method reads jobRcd and calculates the Average Turnover Time
85 * @Param ArrayList<Integer><Integer> jobRcd, job record which makes a work log
86 * @Return double aveTrnovrTime, average turn over time.
87 */
88 1. Accept jobRcd
89 2. Set Su <- 0
90 2. Set loop where the iterator iterates the jobRcd's last column
91 3. Check the last column != 0
92 4. Read the same row column 3 // End Time
93 5. Add sum += endTime
94 6. Move to the next row
95 7. Devide the Sum by the number of row
96 8. Return the result

```

- b. Design the program correctness testing cases. Give at least 3 testing cases to test your program, and give the expected correct **average turnaround time** (for each testing case) in order to test the correctness of each algorithm.

<Complete the following table>

| Testing case # | Input<br>(table of jobs with its<br>job# and length  | Expected<br>output for<br>FCFS<br>(√ if Correct<br>after testing in<br>Part 3) | Expected<br>output for<br>SJF<br>(√ if Correct<br>after testing in<br>Part 3) | Expected<br>output for<br>RR-2<br>(√ if Correct<br>after testing in<br>Part 3) | Expected<br>output for<br>RR-5<br>(√ if Correct<br>after testing in<br>Part 3) |
|----------------|--|--|---|--|--|
| 1<br>(5 jobs)  | Job1<br>7<br>Job2<br>18<br>Job3<br>10<br>Job4<br>4<br>Job5<br>12   | ✓  | ✓   | ✓  | ✓  |
| 2<br>(10 jobs) | Job1<br>23<br>Job2<br>8<br>Job3<br>2<br>Job4<br>24<br>Job5<br>14<br>Job6<br>8<br>Job7<br>27<br>Job8<br>28<br>Job9<br>10<br>Job10<br>25 | ✓  | ✓   | ✓  | ✓  |
| 3<br>(15 jobs) | Job1<br>16<br>Job2<br>29   | ✓  | ✓   | ✓  | ✓  |

|  |  |  |  |  |  |
|--|--|--|--|--|--|
|  | Job3<br>9<br>Job4<br>15<br>Job5<br>13<br>Job6<br>8<br>Job7<br>13<br>Job8<br>1<br>Job9<br>3<br>Job10<br>1<br>Job11<br>12<br>Job12<br>4<br>Job13<br>9<br>Job14<br>1<br>Job15<br>22 |  |  |  |  |
|--|--|--|--|--|--|

- c. Design testing strategy for the programs. Discuss about how to generate and structure the randomly generated inputs for experimental study later in Part 3.

*Hint 1: To study the performance evaluation of the four job scheduling algorithms, this project will use three different input sizes, 5 jobs, 10 jobs and 15 jobs. It is the easiest to use a random number generator for generating the inputs. Note that you need to decide the maximum value of job length (use at least 20). However, student should store each data set in various sizes and use the same data set for each job scheduling algorithm.*

*The performance of average Turnaround Time of each input data size (5 jobs, 10 jobs and 15 jobs) can be calculated after an experiment is conducted in 20 trail (with 20 input sets of jobs). We can denote the results as the set  $X$  which contains the 20 computed Turnaround Times of 20 trails, where  $X = \{x_1, x_2, x_3 \dots x_{20}\}$ , from the simulator.*

*For each data size (5 jobs, 10 jobs and 15 jobs):*

$$\text{Average Turnaround Time} = \frac{\sum_{i=1}^{20} x_i}{20}$$

*The student should decide the maximum value of the job length (at least 20).*

<Insert answers here>

The preprocessing of the four algorithms' execution is generating job data samples. There are 3 groups: 5 jobs, 10 jobs, and 20 jobs groups. Each group has 20 samples. The file contains JobID and the time length of jobs between 1 and 30. After generating a total of 60 samples, the program extracts data from files and executes the four different algorithms, records the average turnover time for each sample, and records the result on CSV files. The last row has the mean of the average turnover time.

**Part 2**  
**Implementation (45 points)**

- a. Code each program based on the design (pseudocode or flow chart) in Part 1(a).

<Generate four programs and stored them in four files, needed to be submitted>

Please see submitted java files.

- b. Document the program appropriately.

<Generate documentation inside the four program files>

Please see submitted java files.

- c. Test you program using the designed testing input data given in the table in Part 1(b), Make sure each program generates the correct answer by marking a “√” if it is correct for each testing case for each program column in the table. Repeat the process of debugging if necessary.

<Complete the four columns of the four algorithms in the table @Part 1(b)>

- d. For each program, capture a screen shot of the execution (Compile&Run) using the testing case in Part 1(b) to show how this program works properly

<Insert totally four screen shots, one for each program, here>

```

SimulatingJobScheduler - Main.java
Project: SimulatingJobScheduler ~./Dropbox
  > .idea
  > AttCSVFiles
  > JobFiles
  > out
  > src
    > JobAlgorithms
      > FCFS
      > FCFSTest
      > RR2
      > RR2Test
      > RR5
      > RR5Test
      > SJF
      > SJFTest
    > WriteFiles
      > WriteCSVFiles
      > WriteCSVFilesTest
    > Main
  > .gitignore
  > SimulatingJobScheduler.iml
  > External Libraries
  > Scratches and Consoles

36 } // end of Main
37
38 /*Sample Run */
39 /*
40
41
42 ~~~15Jobs~~~
43
44
45
46 ~~~~~ File 1 ~~~~~
47
48 --- FCFS ---
49
50 Job ID  Start Time  End Time  Job Completion
51 1      0          16      Job 1completed @ 16
52 2      16         45      Job 2completed @ 45
53 3      45         54      Job 3completed @ 54
54 4      54         69      Job 4completed @ 69
55 5      69         82      Job 5completed @ 82
56 6      82         90      Job 6completed @ 90
57 7      90         103     Job 7completed @ 103
58 8      103        104     Job 8completed @ 104
59 9      104        107     Job 9completed @ 107
60 10     107        108     Job 10completed @ 108
61 11     108        120     Job 11completed @ 120
62 12     120        124     Job 12completed @ 124
63 13     124        133     Job 13completed @ 133
64 14     133        134     Job 14completed @ 134
65 15     134        156     Job 15completed @ 156
66
67
68 Average Turnover Time: 96.3333
69

```

```

71 --- SJF ---
72 Job ID  Start Time  End Time  Job Completion
73 8      0          1      Job 8completed @ 1
74 10     1          2      Job 10completed @ 2
75 14     2          3      Job 14completed @ 3
76 9      3          6      Job 9completed @ 6
77 12     6          10     Job 12completed @ 10
78 6      10         18     Job 6completed @ 18
79 3      18         27     Job 3completed @ 27
80 13     27         36     Job 13completed @ 36
81 11     36         48     Job 11completed @ 48
82 5      48         61     Job 5completed @ 61
83 7      61         74     Job 7completed @ 74
84 4      74         89     Job 4completed @ 89
85 1      89         105     Job 1completed @ 105
86 15     105        127     Job 15completed @ 127
87 2      127        156     Job 2completed @ 156
88
89 Average Turnover Time: 50.8667
90

```

|     |             |            |          |                      |  |     |                                |     |     |                       |
|-----|-------------|------------|----------|----------------------|--|-----|--------------------------------|-----|-----|-----------------------|
| 92  | --- RR2 --- |            |          |                      |  | 131 | 1                              | 70  | 72  |                       |
| 93  | Job ID      | Start Time | End Time | Job Completion       |  | 132 | 2                              | 72  | 74  |                       |
| 94  | 1           | 0          | 2        |                      |  | 133 | 3                              | 74  | 76  |                       |
| 95  | 2           | 2          | 4        |                      |  | 134 | 4                              | 76  | 78  |                       |
| 96  | 3           | 4          | 6        |                      |  | 135 | 5                              | 78  | 80  |                       |
| 97  | 4           | 6          | 8        |                      |  | 136 | 6                              | 80  | 82  | Job 6completed @ 82   |
| 98  | 5           | 8          | 10       |                      |  | 137 | 7                              | 82  | 84  |                       |
| 99  | 6           | 10         | 12       |                      |  | 138 | 11                             | 84  | 86  |                       |
| 100 | 7           | 12         | 14       |                      |  | 139 | 13                             | 86  | 88  |                       |
| 101 | 8           | 14         | 15       | Job 8completed @ 15  |  | 140 | 15                             | 88  | 90  |                       |
| 102 | 9           | 15         | 17       |                      |  | 141 | 1                              | 90  | 92  |                       |
| 103 | 10          | 17         | 18       | Job 10completed @ 18 |  | 142 | 2                              | 92  | 94  |                       |
| 104 | 11          | 18         | 20       |                      |  | 143 | 3                              | 94  | 95  | Job 3completed @ 95   |
| 105 | 12          | 20         | 22       |                      |  | 144 | 4                              | 95  | 97  |                       |
| 106 | 13          | 22         | 24       |                      |  | 145 | 5                              | 97  | 99  |                       |
| 107 | 14          | 24         | 25       | Job 14completed @ 25 |  | 146 | 7                              | 99  | 101 |                       |
| 108 | 15          | 25         | 27       |                      |  | 147 | 11                             | 101 | 103 |                       |
| 109 | 1           | 27         | 29       |                      |  | 148 | 13                             | 103 | 104 | Job 13completed @ 104 |
| 110 | 2           | 29         | 31       |                      |  | 149 | 15                             | 104 | 106 |                       |
| 111 | 3           | 31         | 33       |                      |  | 150 | 1                              | 106 | 108 |                       |
| 112 | 4           | 33         | 35       |                      |  | 151 | 2                              | 108 | 110 |                       |
| 113 | 5           | 35         | 37       |                      |  | 152 | 4                              | 110 | 112 |                       |
| 114 | 6           | 37         | 39       |                      |  | 153 | 5                              | 112 | 114 |                       |
| 115 | 7           | 39         | 41       |                      |  | 154 | 7                              | 114 | 116 |                       |
| 116 | 9           | 41         | 42       | Job 9completed @ 42  |  | 155 | 11                             | 116 | 118 | Job 11completed @ 118 |
| 117 | 11          | 42         | 44       |                      |  | 156 | 15                             | 118 | 120 |                       |
| 118 | 12          | 44         | 46       | Job 12completed @ 46 |  | 157 | 1                              | 120 | 122 |                       |
| 119 | 13          | 46         | 48       |                      |  | 158 | 2                              | 122 | 124 |                       |
| 120 | 15          | 48         | 50       |                      |  | 159 | 4                              | 124 | 126 |                       |
| 121 | 1           | 50         | 52       |                      |  | 160 | 5                              | 126 | 127 | Job 5completed @ 127  |
| 122 | 2           | 52         | 54       |                      |  | 161 | 7                              | 127 | 128 | Job 7completed @ 128  |
| 123 | 3           | 54         | 56       |                      |  | 162 | 15                             | 128 | 130 |                       |
| 124 | 4           | 56         | 58       |                      |  | 163 | 1                              | 130 | 132 | Job 1completed @ 132  |
| 125 | 5           | 58         | 60       |                      |  | 164 | 2                              | 132 | 134 |                       |
| 126 | 6           | 60         | 62       |                      |  | 165 | 4                              | 134 | 135 | Job 4completed @ 135  |
| 127 | 7           | 62         | 64       |                      |  | 166 | 15                             | 135 | 137 |                       |
| 128 | 11          | 64         | 66       |                      |  | 167 | 2                              | 137 | 139 |                       |
| 129 | 13          | 66         | 68       |                      |  | 168 | 15                             | 139 | 141 |                       |
| 130 | 15          | 68         | 70       |                      |  | 169 | 2                              | 141 | 143 |                       |
|     |             |            |          |                      |  | 170 | 15                             | 143 | 145 |                       |
|     |             |            |          |                      |  | 171 | 2                              | 145 | 147 |                       |
|     |             |            |          |                      |  | 172 | 15                             | 147 | 149 | Job 15completed @ 149 |
|     |             |            |          |                      |  | 173 | 2                              | 149 | 151 |                       |
|     |             |            |          |                      |  | 174 | 2                              | 151 | 153 |                       |
|     |             |            |          |                      |  | 175 | 2                              | 153 | 155 |                       |
|     |             |            |          |                      |  | 176 | 2                              | 155 | 156 | Job 2completed @ 156  |
|     |             |            |          |                      |  | 177 |                                |     |     |                       |
|     |             |            |          |                      |  | 178 | Average Turnover Time: 91.4667 |     |     |                       |
|     |             |            |          |                      |  | 179 |                                |     |     |                       |

|     |                                |            |          |                       |
|-----|--------------------------------|------------|----------|-----------------------|
| 181 | --- RR5 ---                    |            |          |                       |
| 182 | Job ID                         | Start Time | End Time | Job Completion        |
| 183 | 1                              | 0          | 5        |                       |
| 184 | 2                              | 5          | 10       |                       |
| 185 | 3                              | 10         | 15       |                       |
| 186 | 4                              | 15         | 20       |                       |
| 187 | 5                              | 20         | 25       |                       |
| 188 | 6                              | 25         | 30       |                       |
| 189 | 7                              | 30         | 35       |                       |
| 190 | 8                              | 35         | 36       | Job 8completed @ 36   |
| 191 | 9                              | 36         | 39       | Job 9completed @ 39   |
| 192 | 10                             | 39         | 40       | Job 10completed @ 40  |
| 193 | 11                             | 40         | 45       |                       |
| 194 | 12                             | 45         | 49       | Job 12completed @ 49  |
| 195 | 13                             | 49         | 54       |                       |
| 196 | 14                             | 54         | 55       | Job 14completed @ 55  |
| 197 | 15                             | 55         | 60       |                       |
| 198 | 1                              | 60         | 65       |                       |
| 199 | 2                              | 65         | 70       |                       |
| 200 | 3                              | 70         | 74       | Job 3completed @ 74   |
| 201 | 4                              | 74         | 79       |                       |
| 202 | 5                              | 79         | 84       |                       |
| 203 | 6                              | 84         | 87       | Job 6completed @ 87   |
| 204 | 7                              | 87         | 92       |                       |
| 205 | 11                             | 92         | 97       |                       |
| 206 | 13                             | 97         | 101      | Job 13completed @ 101 |
| 207 | 15                             | 101        | 106      |                       |
| 208 | 1                              | 106        | 111      |                       |
| 209 | 2                              | 111        | 116      |                       |
| 210 | 4                              | 116        | 121      | Job 4completed @ 121  |
| 211 | 5                              | 121        | 124      | Job 5completed @ 124  |
| 212 | 7                              | 124        | 127      | Job 7completed @ 127  |
| 213 | 11                             | 127        | 129      | Job 11completed @ 129 |
| 214 | 15                             | 129        | 134      |                       |
| 215 | 1                              | 134        | 135      | Job 1completed @ 135  |
| 216 | 2                              | 135        | 140      |                       |
| 217 | 15                             | 140        | 145      |                       |
| 218 | 2                              | 145        | 150      |                       |
| 219 | 15                             | 150        | 152      | Job 15completed @ 152 |
| 220 | 2                              | 152        | 156      | Job 2completed @ 156  |
| 221 |                                |            |          |                       |
| 222 | Average Turnover Time: 95.0000 |            |          |                       |
| 223 |                                |            |          |                       |

By now, four working programs are created and ready for experimental study in the next part, Part 3.



**Part 3**  
**Performance Analysis (60 points)**

- a. Run each program with the designed randomly generated input data given in Part 1(c).  
Generate a table for all the experimental results for performance analysis as follows.

| Input<br>Size<br>n jobs | Average of average<br>turnaround times<br>(FCFS Program) | Average of average<br>turnaround times<br>(SFJ Program) | Average of average<br>turnaround times<br>(RR-2) | Average of average<br>turnaround times<br>(RR-5) |
|-------------------------|--|---|--|--|
| 5 jobs                  | 43.5   | 35.54   | 54.56  | 53.62  |
| 10 jobs                 | 83.255   | 62.175  | 106.09   | 104.68   |
| 15 jobs                 | 123.183333   | 88.7833333  | 157.69   | 157.136667                                       |

5Jobs Average Turnover Time table.

| File #   | FCFS | SJF   | RR2   | RR5   |
|----------|------|-------|-------|-------|
| 1        | 31.4 | 24    | 36.4  | 36    |
| 2        | 45.2 | 35.2  | 54.8  | 55.2  |
| 3        | 43.2 | 38.6  | 59.4  | 61.6  |
| 4        | 40.8 | 34    | 50.4  | 49    |
| 5        | 26.8 | 26    | 37.2  | 32    |
| 6        | 63   | 47.6  | 73.8  | 73.8  |
| 7        | 34.8 | 17    | 24.2  | 24.4  |
| 8        | 24.2 | 23.6  | 32.8  | 31.2  |
| 9        | 55   | 50.8  | 79.6  | 79    |
| 10       | 21.2 | 15.6  | 21.6  | 20.8  |
| 11       | 27.6 | 24.4  | 34    | 32.6  |
| 12       | 47.2 | 33.2  | 50.4  | 50.2  |
| 13       | 51   | 49    | 78.2  | 74.4  |
| 14       | 55.8 | 51.6  | 80.8  | 77    |
| 15       | 51   | 30.6  | 45.8  | 45.6  |
| 16       | 49   | 44    | 68.4  | 63    |
| 17       | 76.8 | 68.4  | 112.4 | 113.6 |
| 18       | 43.6 | 30.4  | 48    | 48    |
| 19       | 24.6 | 15    | 21.6  | 24.2  |
| 20       | 57.8 | 51.8  | 81.4  | 80.8  |
| mean ATT | 43.5 | 35.54 | 54.56 | 53.62 |

10Jobs Average Turnover Time table.

| File #   | FCFS   | SJF    | RR2    | RR5    |
|----------|--------|--------|--------|--------|
| 1        | 84.7   | 67.8   | 113.4  | 111.3  |
| 2        | 87.4   | 60.3   | 102.2  | 104.1  |
| 3        | 85.4   | 80.3   | 136.8  | 131    |
| 4        | 65.7   | 53.3   | 87.7   | 87.3   |
| 5        | 71.7   | 40.9   | 69.6   | 70.2   |
| 6        | 90.7   | 79.4   | 137.6  | 134.8  |
| 7        | 87.1   | 49.1   | 83.5   | 83.5   |
| 8        | 59.9   | 39.4   | 67.4   | 71     |
| 9        | 94.7   | 64.9   | 112.3  | 109.9  |
| 10       | 71.7   | 55.6   | 93.4   | 92.5   |
| 11       | 90.6   | 84.4   | 146.5  | 141.3  |
| 12       | 86.7   | 56.1   | 96.2   | 95.7   |
| 13       | 47.6   | 31.5   | 49.2   | 48.6   |
| 14       | 66     | 49.6   | 82.2   | 78     |
| 15       | 90.7   | 67.2   | 114.8  | 112.5  |
| 16       | 106.2  | 81.3   | 140.7  | 139.7  |
| 17       | 79     | 60.3   | 102.2  | 98.5   |
| 18       | 86.2   | 62.2   | 106.5  | 104.2  |
| 19       | 98.5   | 73.6   | 127.7  | 126.4  |
| 20       | 114.6  | 86.3   | 151.9  | 153.1  |
| mean ATT | 83.255 | 62.175 | 106.09 | 104.68 |

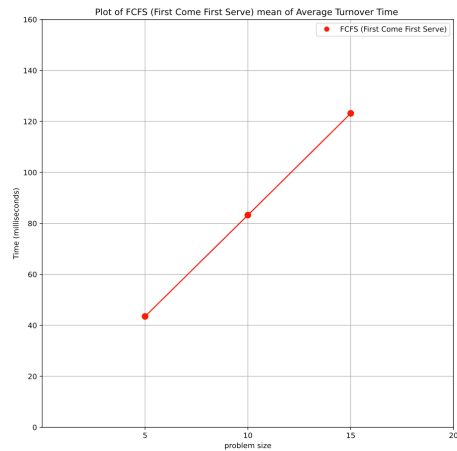
15Jobs Average Turnover Time table.

| File #   | FCFS       | SJF        | RR2        | RR5        |
|----------|------------|------------|------------|------------|
| 1        | 96.3333333 | 50.8666667 | 91.4666667 | 95         |
| 2        | 84.4       | 58.0666667 | 97.8       | 98.3333333 |
| 3        | 135.6      | 97.5333333 | 173.066667 | 178.133333 |
| 4        | 150.4      | 104.133333 | 187.066667 | 188.8      |
| 5        | 115.4      | 90.2       | 158.4      | 157.6      |
| 6        | 112.266667 | 67.5333333 | 118.066667 | 117.133333 |
| 7        | 121.933333 | 99.0666667 | 175.6      | 179.066667 |
| 8        | 118.333333 | 86.6       | 153.133333 | 149        |
| 9        | 113.866667 | 91.5333333 | 162.666667 | 159.333333 |
| 10       | 134.533333 | 87.1333333 | 155.6      | 162.2      |
| 11       | 135        | 93.8       | 169.666667 | 166.8      |
| 12       | 124.6      | 104.466667 | 186.6      | 183.466667 |
| 13       | 157.466667 | 103.933333 | 187.666667 | 189        |
| 14       | 127.666667 | 99.6       | 177        | 173        |
| 15       | 158.466667 | 117.133333 | 214.333333 | 210.8      |
| 16       | 118.8      | 70.4       | 121.866667 | 127.866667 |
| 17       | 112.266667 | 101.2      | 178.2      | 170.2      |
| 18       | 131.4      | 99.9333333 | 180.066667 | 178.066667 |
| 19       | 103.733333 | 68.4666667 | 119.133333 | 116.666667 |
| 20       | 111.2      | 84.0666667 | 146.4      | 142.266667 |
| mean ATT | 123.183333 | 88.7833333 | 157.69     | 157.136667 |

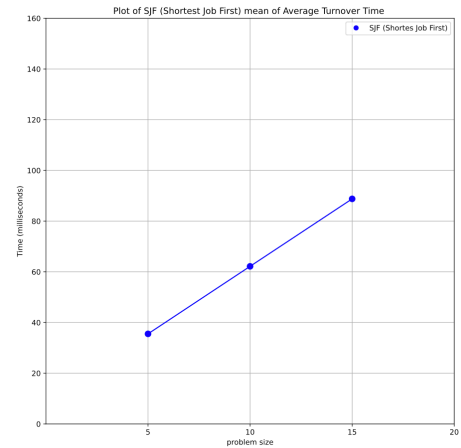
- b. Plot a graph of each algorithm, average turnaround time vs input size (# of jobs), and summarize the performance of each algorithm based on its own graph.

<Insert totally four graphs, one for each program, here>

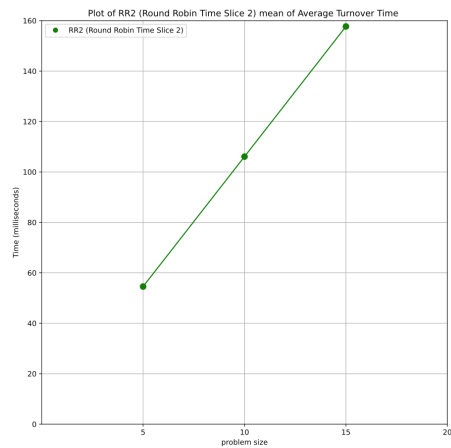
### 1. FCFS



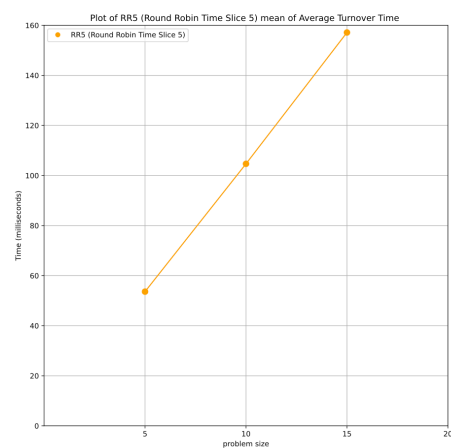
### 2. SJF



### 3. RR2



### 4. RR5



<Write a summary>

**1. First Come First Serve (FCFS):**

As the number of jobs increases from 5 to 15, the mean ATT also increases linearly. It can be expected easily since the queue gets longer, and jobs have to wait more. As we discussed in class, FCFS generally has a higher ATT than Shortest Job First (SJF), indicating that jobs with longer processing times can increase the waiting time for subsequent jobs.

**2. Shortest Job First (SJF):**

It consistently shows the lowest mean ATT across all scenarios as discussed in class. It reduces the amount of time shorter jobs have to wait. The growing execution time is the slowest among the four algorithms. It may imply SJF is effective when there are many short jobs mixed with longer jobs, as it minimizes the overall waiting time.

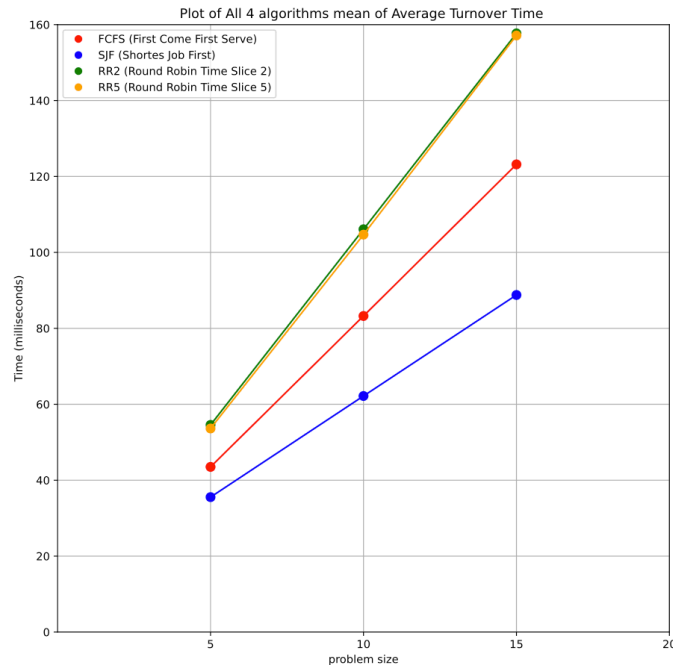
**3. Round Robin (RR) with Time Slice 2:**

The mean ATT is higher compared to SJF and FCFS, especially as the number of jobs increases. It can be due to the overhead of context switching and the fact that longer jobs will take more cycles to complete. The time slice of 2 might be too short for some jobs, leading to increased overhead.

**4. Round Robin (RR) with Time Slice 5:**

I expected RR5 would be more efficient than RR2 with a larger time slice. Jobs have more time to execute before being preempted, which can be more efficient for longer jobs. But it can also lead to longer waiting times for shorter jobs if they arrive behind longer jobs. The mean ATT for RR with a time slice of 5 is slightly lower than that of a time slice of 2, indicating a possible reduction in context switching overhead. However, it's still higher than SJF and FCFS.

Plot all four graphs on the same graph and compare the performance of all four algorithms. Rank four scheduling algorithms. Try giving the reasons for the findings.  
 <Insert four-graphs-in-one graph here> <Write about explaining the results>



From the observations above and the comparison four algorithms' execution time on the same plot, we can infer the following: SJF is the most efficient in terms of average turnaround time across the board. It may be effective when many short jobs are mixed with longer ones. FCFS is predictable, and short-term jobs would be stuck waiting behind long-term jobs. Round Robin is generally fairer regarding CPU time distribution but can have higher average turnaround times. But we cannot see much difference between time slices 2 and 5.

RR2 and RR5 show a steeper ATT increase compared to SJF and FCFS. This suggests that as the problem size increases, the effect of context switching causes inefficiency in handling longer jobs and higher ATT. Comparing RR2 and RR5, we can see that RR5 generally has a slightly lower ATT than RR2. It might come from the longer time slice in RR5, which reduces the frequency of context switches, which can be particularly costly in terms of time, and improves the ATT slightly. However, both versions of Round Robin have higher ATT than SJF and FCFS as the problem size increases.

As the problem size grows from 5 to 15, the gap in performance between the algorithms also increases. It shows that the choice of algorithm becomes more critical as the number of jobs scales up. The consistent performance of SJF across different problem sizes highlights its efficiency.

The plot shows that SJF is the most efficient algorithm in this circumstance. However, the result highly depends on the experimental environment, such as hardware components, number of jobs, and time length of jobs. The choice of scheduling algorithm should be based on the specific characteristics of the workload.

- c. Conclude your report with the strength and constraints of your work. At least 100 words.

(Note: It is reflection of this project. If you have a change to re-do this project again, what you like to keep and what you like to do differently in order get a better quality of results.)

<Write a conclusion about strength and constraints of your work here.>

Implementing and analyzing the scheduling algorithms discussed in our class has been a significant experience. The plots show that while SJF records are the most efficient in our simulations, their performance may vary in different computational environments or with different job arrival patterns. The project did not include priority-based scheduling algorithms, which could add another interpretation to algorithm analysis.

In future iterations of this project, I would include a variety of round-robin time slices, such as 50 and 100, to examine how the length of time slice affects algorithm performance further. Also, I could generate job batches that consist exclusively of short or long tasks to investigate how job length influences scheduling efficacy. These modifications and varying the mix of job lengths in the input files can project additional insights and enhance our understanding of scheduling algorithm characteristics.