CIE 302: Operating systems
Project Report

Team members:

- ❏ Salma Mahran (201801012)
- ❏ Radwa Elsawy (201800901)

Data structure used:

In the process_generator:

- ❏ Process queue: a FIFO queue that stores the information of the processes after reading them from the input file.

In the scheduler:

- ❏ **Process table:** Static array of processes.
- ❏ **Ready queue :**

- For SJF: a priority queue with a runtime of the process used as the priority.
- For RR: a FIFO queue.
- For preemptive HPF: a priority queue with a priority of the process used to order the process and determine which should be served first.
- For preemptive SRTN: a priority queue with the remaining time of the process used as a priority.

- ❏ **Memory allocation & deallocation:**

1. Leaves queue:
    - using Leaves Queue for storing the free nodes of the binary tree with a priority of the node memory size.
    - It is a priority queue that orders its nodes (pointers on memory blocks) according to their both sizes and starting index of the memory block. it is used to keep track of the empty memory blocks and to facilitate the allocation process.
2. Binary tree:

    - using a binary tree for splitting the memory to the suitable and usable storage, then merging the free nodes after the deallocation.
    - It is used to keep track of the free and allocated memory blocks. It is mainly used to organize the deallocation and the merging of memory spaces as it shows which memory block is free and which is not and enables memory to see whether its neighbors are free to merge or not.

Description of project:

Algorithm explanation and results:

In all the algorithms, the scheduler should perform two main functions

1- when receiving a signal from the process generator, it should fork the process and add it to the ready queue.

2- choose which process will be allocated the CPU. This happens inside scheduling_SJF, scheduling_RR, or scheduling_HPF according to the scheduling algorithm chosen by the user.

Note:

The scheduler finishes its work and terminates after checking:

1. that the process generator won't send any other process(the process generator sends **SIGUSER2** to the scheduler when it sends all the process and the scheduler receives the signal and marks the event)

2. that the ready queue is empty.

- **scheduling_SJF:**

  The scheduler picks the process at the front of the priority queue, sends a SIGCONT signal so that process continues its work. Then the scheduler waits for the chosen process to finish its work using waitpid() system call. When the process finishes the scheduler starts this loop again.

- **scheduling_RR:**

  The scheduler picks the first process. It compares the quantum and the remaining time for the process and chooses the least to be the rem_sleep (the amount of time that the scheduler will be asleep to give time for the process to run). Then the scheduler sends a SIGCONT signal so that process continues its work and goes to sleep for rem_sleep.

  When the scheduler wakes up, it checks the remaining time of the running process, if it is greater than 0, it sends a SIGSTOP to the process and adds it to the queue. If it was 0, it waits for the SIGCHLD signal from the running process to make sure that it really has finished. Then the scheduler repeats.

- **scheduling_HPF:**

  The scheduler picks the process at the front of the priority queue, sends a SIGCONT signal so that process continues its work. Then the scheduler goes to sleep for an amount of time equals the remaining time of the process until it gets another process with higher priority. If the new process has higher priority then the scheduling will stop working with the current process(Sending SIGSTOP) and start working on the new process.

- **scheduling_SRTN:**

  it works with the same logic as the scheduling_HPF but orders its process according to the remaining time not according to the priority.

- **buddy allocation system:**

  To facilitate the allocation and the deallocation of memory, we use two data structures to keep track of memory blocks: a binary tree and a priority queue (leaves queue). In the beginning, we keep memory space as four memory blocks, each has a size of 256, in the leaves_queue. When a new process is sent to the scheduler, the scheduler, inside the allocate function, searches for the smallest memory block that will hold the process and take this node out of the leaves queue. then, it checks if the size of this memory block is more or equal to the size of the process, if so the memory block will be split and then it will recheck the same condition again until it no longer holds. And all that gets updated in the binary tree, any new size free node and the allocated nodes are implemented in the tree. when a process finishes, its corresponding memory block is deallocated. In the scheduler, the deallocation function checks if the neighbor memory block is free to merge or not. If yes, it merges the two nodes into a parent node with twice the size. Then it keeps doing that until the nodes can not be merged anymore. After that, it takes the pointer on the memory block and pushes it back with its size into the leaves queue.

Assumptions:

- In Round Robin: the process, new one or stopped one, that entered the ready queue will start working first.

-In HPF:

- If it gets two new processes with the same priority with different arrival times, then the last entered process in the priority queue will start working first.
- If it gets a process with the same priority as the running process, it will keep the running process until it finishes, then start the new one.

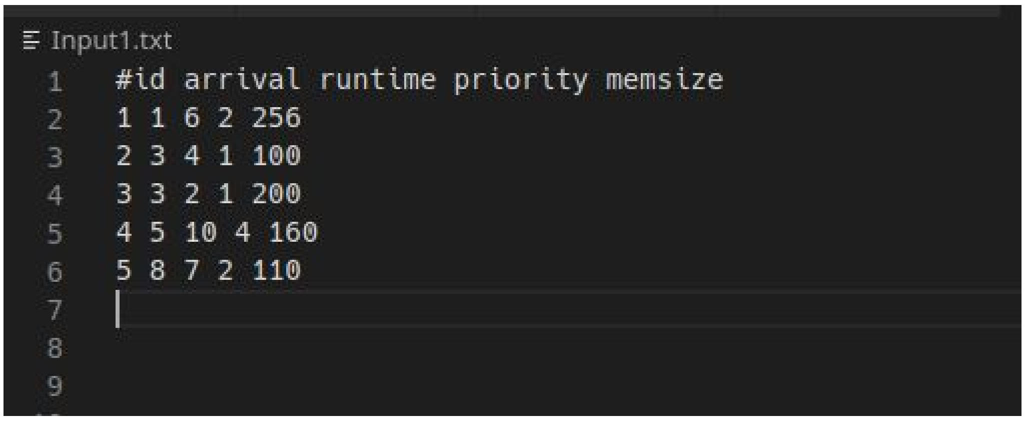
-In SRTN: the same assumptions of HPF.

-For all schedulers:

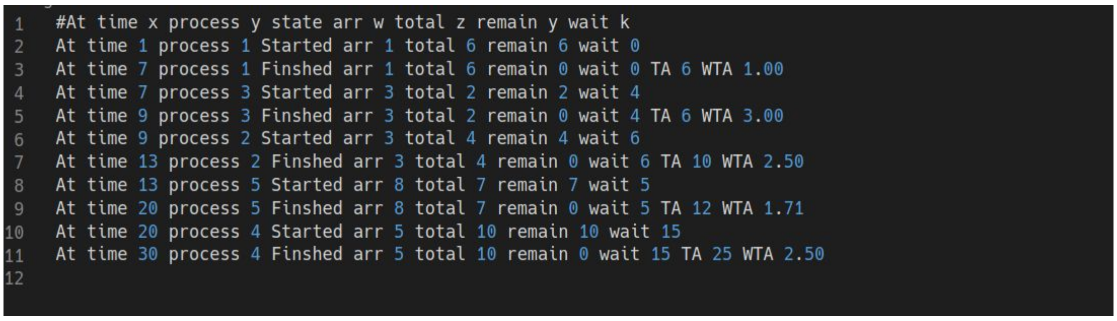
The allocation function takes part before the deallocation. So, if there are no free nodes, then any new process must arrive at least one-time step after the finished time of any running process. (Can't allocate any process before that).

Test cases and Screenshots:

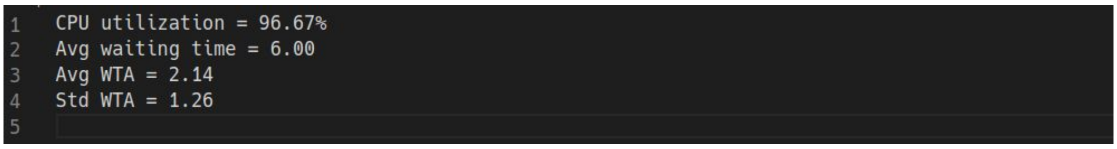
SJF Case1:



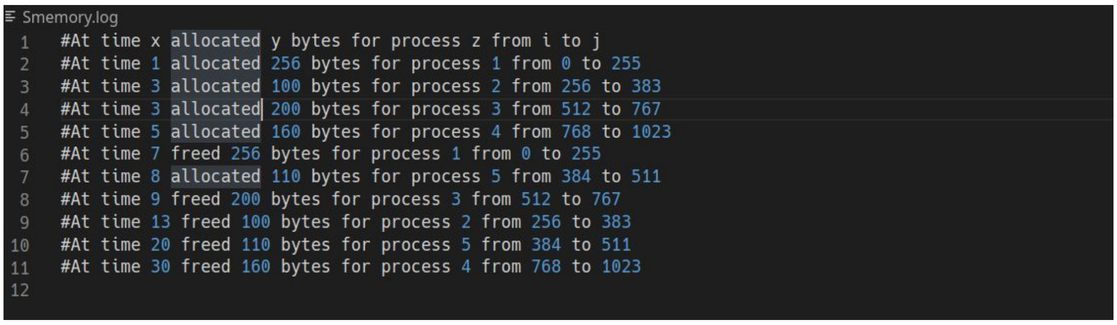
```
Input1.txt
#id arrival runtime priority memsize
1 1 6 2 256
2 3 4 1 100
3 3 2 1 200
4 5 10 4 160
5 8 7 2 110
```



```
#At time x process y state arr w total z remain y wait k
At time 1 process 1 Started arr 1 total 6 remain 6 wait 0
At time 7 process 1 Finshed arr 1 total 6 remain 0 wait 0 TA 6 WTA 1.00
At time 7 process 3 Started arr 3 total 2 remain 2 wait 4
At time 9 process 3 Finshed arr 3 total 2 remain 0 wait 4 TA 6 WTA 3.00
At time 9 process 2 Started arr 3 total 4 remain 4 wait 6
At time 13 process 2 Finshed arr 3 total 4 remain 0 wait 6 TA 10 WTA 2.50
At time 13 process 5 Started arr 8 total 7 remain 7 wait 5
At time 20 process 5 Finshed arr 8 total 7 remain 0 wait 5 TA 12 WTA 1.71
At time 20 process 4 Started arr 5 total 10 remain 10 wait 15
At time 30 process 4 Finshed arr 5 total 10 remain 0 wait 15 TA 25 WTA 2.50
```

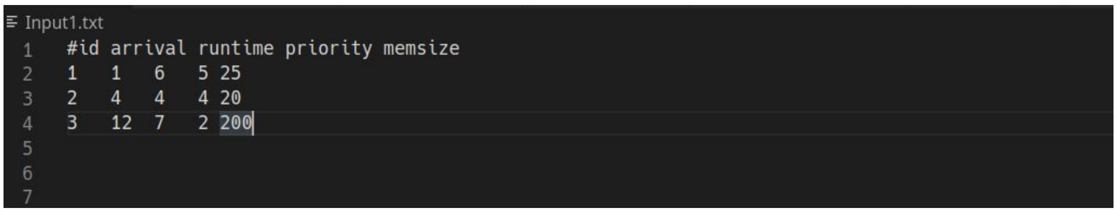


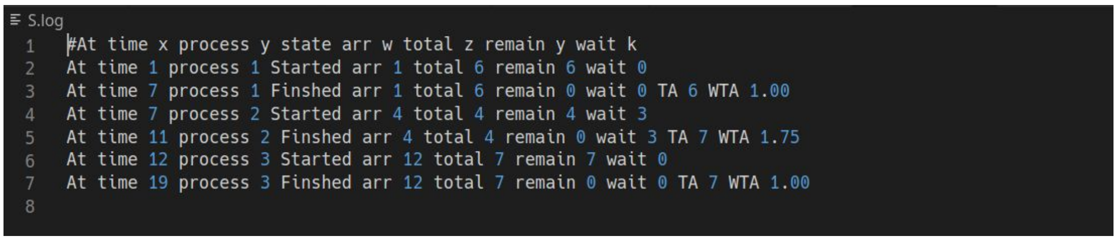
```
CPU utilization = 96.67%
Avg waiting time = 6.00
Avg WTA = 2.14
Std WTA = 1.26
```

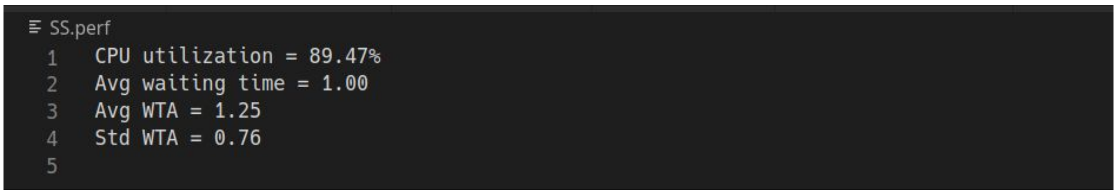


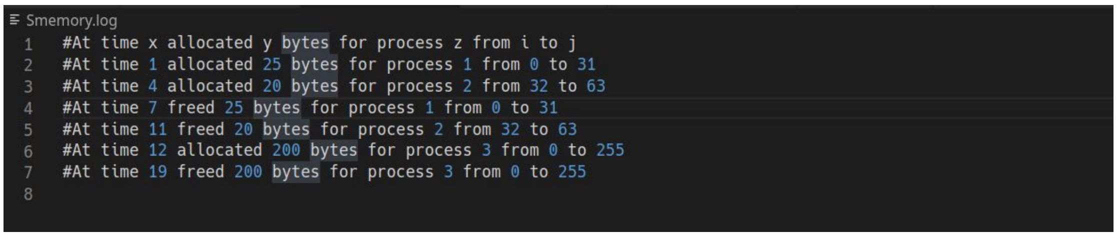
```
Smemory.log
#At time x allocated y bytes for process z from i to j
#At time 1 allocated 256 bytes for process 1 from 0 to 255
#At time 3 allocated 100 bytes for process 2 from 256 to 383
#At time 3 allocated 200 bytes for process 3 from 512 to 767
#At time 5 allocated 160 bytes for process 4 from 768 to 1023
#At time 7 freed 256 bytes for process 1 from 0 to 255
#At time 8 allocated 110 bytes for process 5 from 384 to 511
#At time 9 freed 200 bytes for process 3 from 512 to 767
#At time 13 freed 100 bytes for process 2 from 256 to 383
#At time 20 freed 110 bytes for process 5 from 384 to 511
#At time 30 freed 160 bytes for process 4 from 768 to 1023
```

SJF Case2:

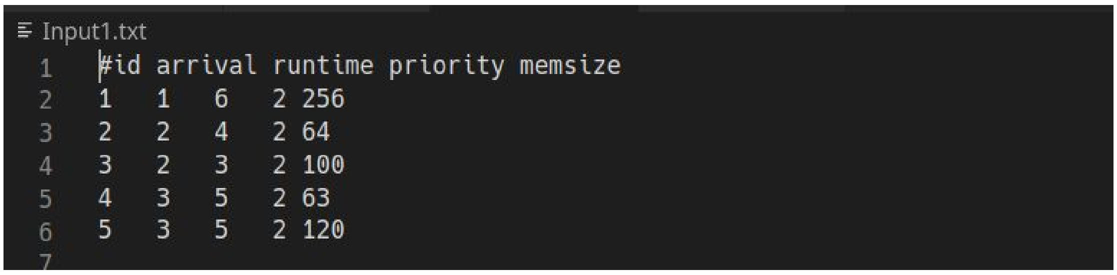

Input1.txt
```
#id arrival runtime priority memsize
1    1    6    5 25
2    4    4    4 20
3    12   7    2 200
```

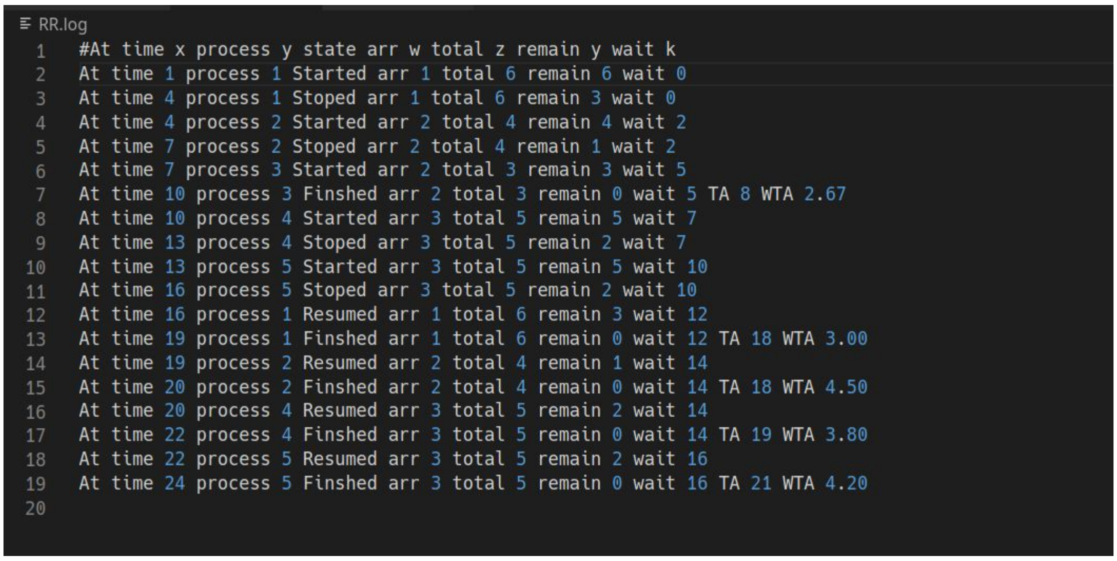

S.log
```
#At time x process y state arr w total z remain y wait k
At time 1 process 1 Started arr 1 total 6 remain 6 wait 0
At time 7 process 1 Finshed arr 1 total 6 remain 0 wait 0 TA 6 WTA 1.00
At time 7 process 2 Started arr 4 total 4 remain 4 wait 3
At time 11 process 2 Finshed arr 4 total 4 remain 0 wait 3 TA 7 WTA 1.75
At time 12 process 3 Started arr 12 total 7 remain 7 wait 0
At time 19 process 3 Finshed arr 12 total 7 remain 0 wait 0 TA 7 WTA 1.00
```


SS.perf
```
CPU utilization = 89.47%
Avg waiting time = 1.00
Avg WTA = 1.25
Std WTA = 0.76
```

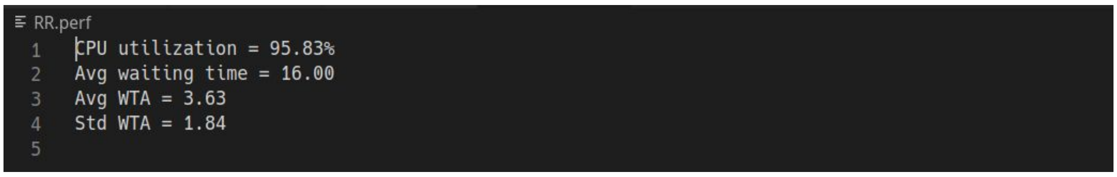

Smemory.log
```
#At time x allocated y bytes for process z from i to j
#At time 1 allocated 25 bytes for process 1 from 0 to 31
#At time 4 allocated 20 bytes for process 2 from 32 to 63
#At time 7 freed 25 bytes for process 1 from 0 to 31
#At time 11 freed 20 bytes for process 2 from 32 to 63
#At time 12 allocated 200 bytes for process 3 from 0 to 255
#At time 19 freed 200 bytes for process 3 from 0 to 255
```

RR case 1 (quantum 3):

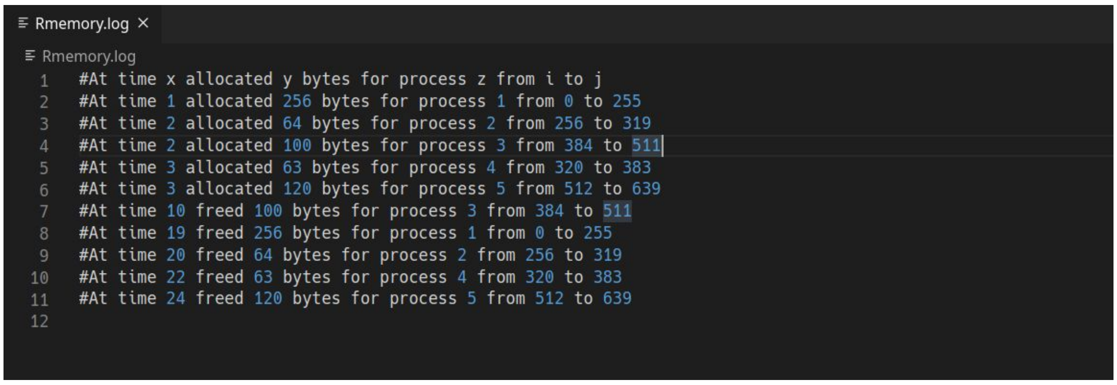

Input1.txt
```
#id arrival runtime priority memsize
1    1    6    2 256
2    2    4    2 64
3    2    3    2 100
4    3    5    2 63
5    3    5    2 120
```


RR.log

```
#At time x process y state arr w total z remain y wait k
At time 1 process 1 Started arr 1 total 6 remain 6 wait 0
At time 4 process 1 Stoped arr 1 total 6 remain 3 wait 0
At time 4 process 2 Started arr 2 total 4 remain 4 wait 2
At time 7 process 2 Stoped arr 2 total 4 remain 1 wait 2
At time 7 process 3 Started arr 2 total 3 remain 3 wait 5
At time 10 process 3 Finshed arr 2 total 3 remain 0 wait 5 TA 8 WTA 2.67
At time 10 process 4 Started arr 3 total 5 remain 5 wait 7
At time 13 process 4 Stoped arr 3 total 5 remain 2 wait 7
At time 13 process 5 Started arr 3 total 5 remain 5 wait 10
At time 16 process 5 Stoped arr 3 total 5 remain 2 wait 10
At time 16 process 1 Resumed arr 1 total 6 remain 3 wait 12
At time 19 process 1 Finshed arr 1 total 6 remain 0 wait 12 TA 18 WTA 3.00
At time 19 process 2 Resumed arr 2 total 4 remain 1 wait 14
At time 20 process 2 Finshed arr 2 total 4 remain 0 wait 14 TA 18 WTA 4.50
At time 20 process 4 Resumed arr 3 total 5 remain 2 wait 14
At time 22 process 4 Finshed arr 3 total 5 remain 0 wait 14 TA 19 WTA 3.80
At time 22 process 5 Resumed arr 3 total 5 remain 2 wait 16
At time 24 process 5 Finshed arr 3 total 5 remain 0 wait 16 TA 21 WTA 4.20
```


RR.perf

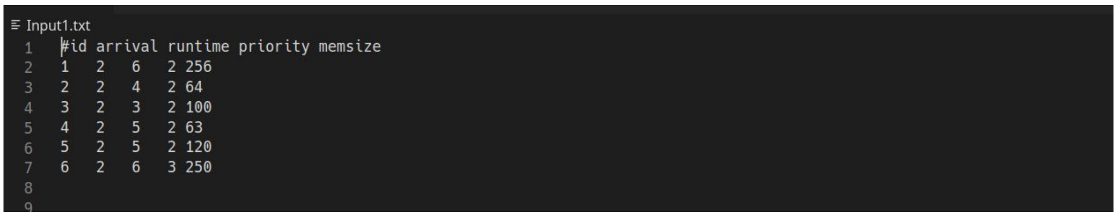
```
CPU utilization = 95.83%
Avg waiting time = 16.00
Avg WTA = 3.63
Std WTA = 1.84
```


Rmemory.log ×

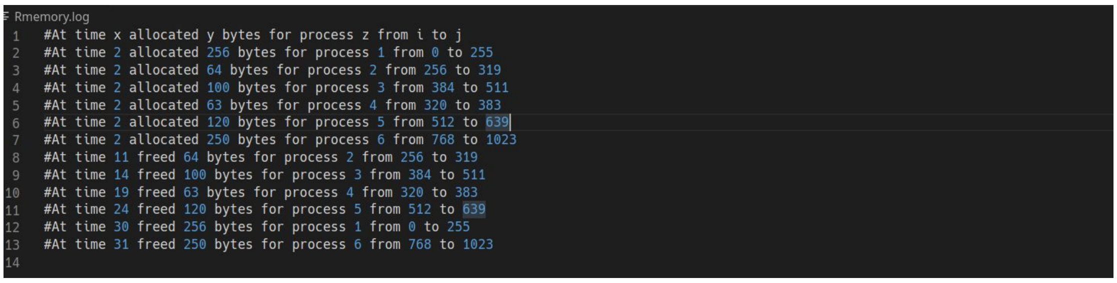
Rmemory.log

```
#At time x allocated y bytes for process z from i to j
#At time 1 allocated 256 bytes for process 1 from 0 to 255
#At time 2 allocated 64 bytes for process 2 from 256 to 319
#At time 2 allocated 100 bytes for process 3 from 384 to 511
#At time 3 allocated 63 bytes for process 4 from 320 to 383
#At time 3 allocated 120 bytes for process 5 from 512 to 639
#At time 10 freed 100 bytes for process 3 from 384 to 511
#At time 19 freed 256 bytes for process 1 from 0 to 255
#At time 20 freed 64 bytes for process 2 from 256 to 319
#At time 22 freed 63 bytes for process 4 from 320 to 383
#At time 24 freed 120 bytes for process 5 from 512 to 639
```

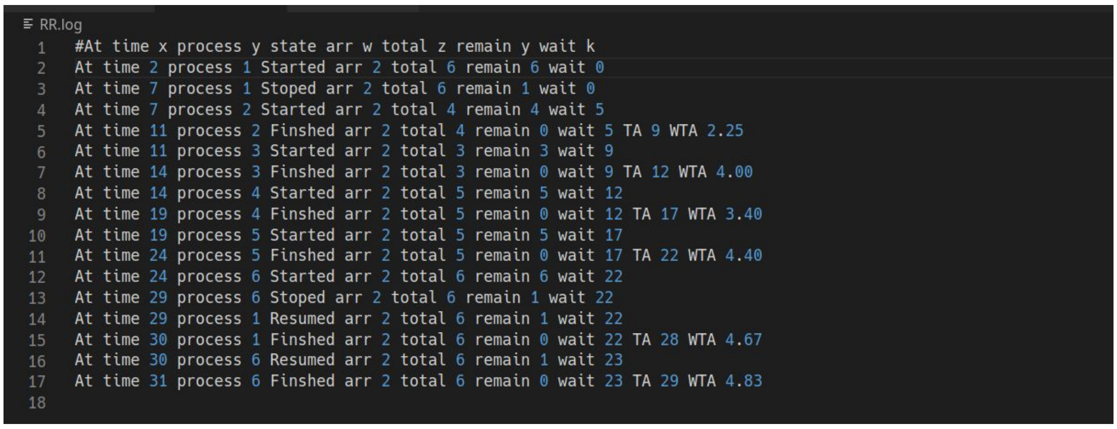
RR case 2 (quantum 5)


Input1.txt

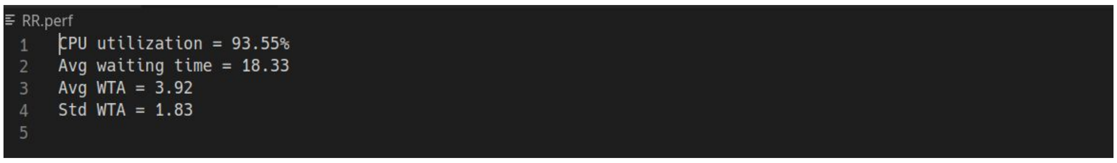
```
#id arrival runtime priority memsize
1   2   6   2 256
2   2   4   2 64
3   2   3   2 100
4   2   5   2 63
5   2   5   2 120
6   2   6   3 250
```


Rmemory.log

```
#At time x allocated y bytes for process z from i to j
#At time 2 allocated 256 bytes for process 1 from 0 to 255
#At time 2 allocated 64 bytes for process 2 from 256 to 319
#At time 2 allocated 100 bytes for process 3 from 384 to 511
#At time 2 allocated 63 bytes for process 4 from 320 to 383
#At time 2 allocated 120 bytes for process 5 from 512 to 639
#At time 2 allocated 250 bytes for process 6 from 768 to 1023
#At time 11 freed 64 bytes for process 2 from 256 to 319
#At time 14 freed 100 bytes for process 3 from 384 to 511
#At time 19 freed 63 bytes for process 4 from 320 to 383
#At time 24 freed 120 bytes for process 5 from 512 to 639
#At time 30 freed 256 bytes for process 1 from 0 to 255
#At time 31 freed 250 bytes for process 6 from 768 to 1023
```

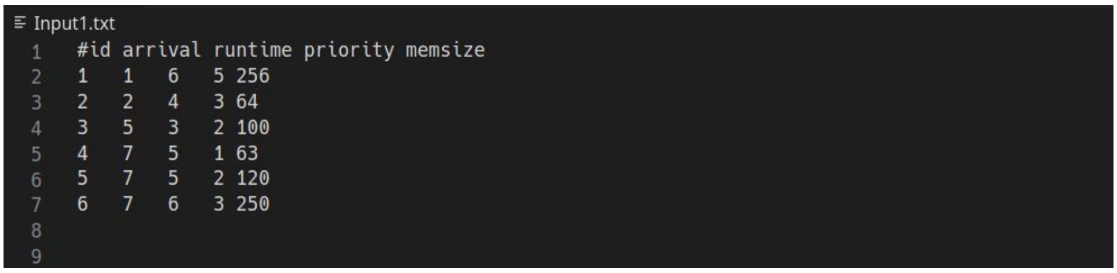


```
#At time x process y state arr w total z remain y wait k
At time 2 process 1 Started arr 2 total 6 remain 6 wait 0
At time 7 process 1 Stoped arr 2 total 6 remain 1 wait 0
At time 7 process 2 Started arr 2 total 4 remain 4 wait 5
At time 11 process 2 Finshed arr 2 total 4 remain 0 wait 5 TA 9 WTA 2.25
At time 11 process 3 Started arr 2 total 3 remain 3 wait 9
At time 14 process 3 Finshed arr 2 total 3 remain 0 wait 9 TA 12 WTA 4.00
At time 14 process 4 Started arr 2 total 5 remain 5 wait 12
At time 19 process 4 Finshed arr 2 total 5 remain 0 wait 12 TA 17 WTA 3.40
At time 19 process 5 Started arr 2 total 5 remain 5 wait 17
At time 24 process 5 Finshed arr 2 total 5 remain 0 wait 17 TA 22 WTA 4.40
At time 24 process 6 Started arr 2 total 6 remain 6 wait 22
At time 29 process 6 Stoped arr 2 total 6 remain 1 wait 22
At time 29 process 1 Resumed arr 2 total 6 remain 1 wait 22
At time 30 process 1 Finshed arr 2 total 6 remain 0 wait 22 TA 28 WTA 4.67
At time 30 process 6 Resumed arr 2 total 6 remain 1 wait 23
At time 31 process 6 Finshed arr 2 total 6 remain 0 wait 23 TA 29 WTA 4.83
```

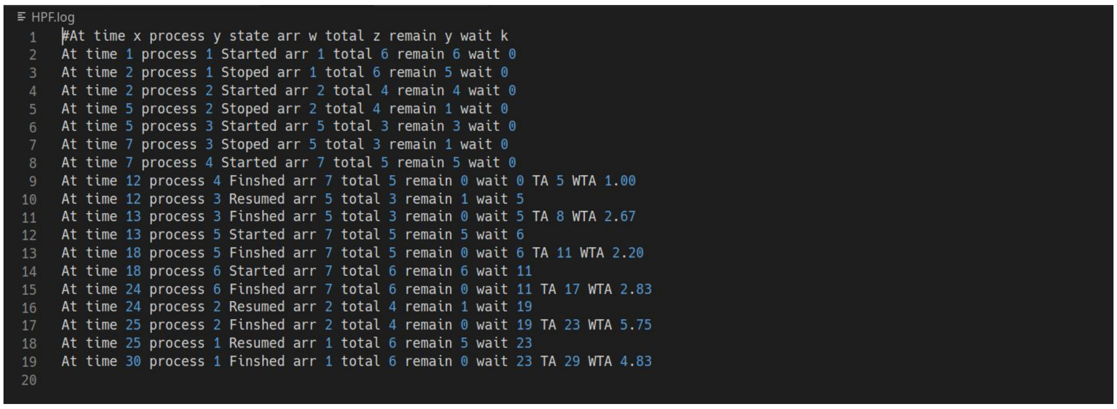


```
CPU utilization = 93.55%
Avg waiting time = 18.33
Avg WTA = 3.92
Std WTA = 1.83
```

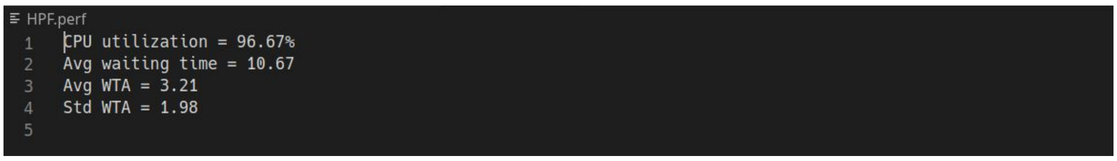
HPF case 1:



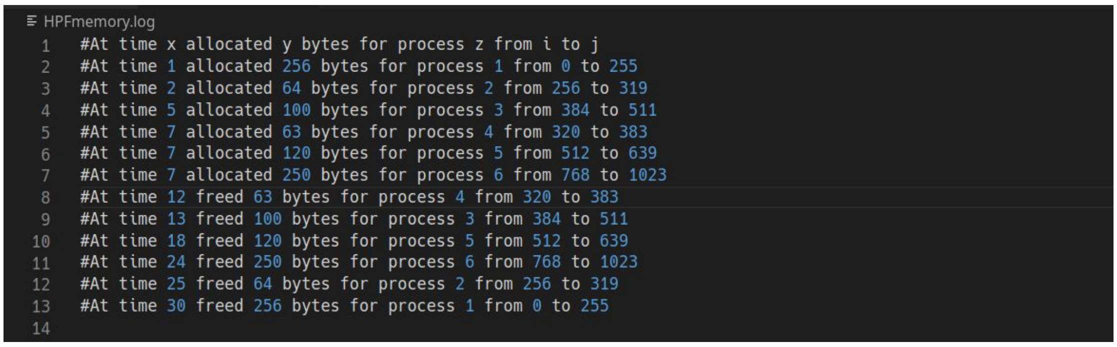
```
#id arrival runtime priority memsize
1   1   6   5 256
2   2   4   3 64
3   5   3   2 100
4   7   5   1 63
5   7   5   2 120
6   7   6   3 250
```



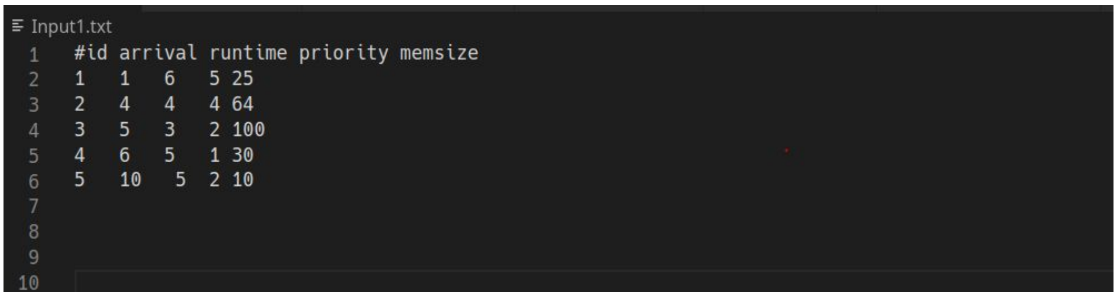
```
#At time x process y state arr w total z remain y wait k
At time 1 process 1 Started arr 1 total 6 remain 6 wait 0
At time 2 process 1 Stoped arr 1 total 6 remain 5 wait 0
At time 2 process 2 Started arr 2 total 4 remain 4 wait 0
At time 5 process 2 Stoped arr 2 total 4 remain 1 wait 0
At time 5 process 3 Started arr 5 total 3 remain 3 wait 0
At time 7 process 3 Stoped arr 5 total 3 remain 1 wait 0
At time 7 process 4 Started arr 7 total 5 remain 5 wait 0
At time 12 process 4 Finshed arr 7 total 5 remain 0 wait 0 TA 5 WTA 1.00
At time 12 process 3 Resumed arr 5 total 3 remain 1 wait 5
At time 13 process 3 Finshed arr 5 total 3 remain 0 wait 5 TA 8 WTA 2.67
At time 13 process 5 Started arr 7 total 5 remain 5 wait 6
At time 18 process 5 Finshed arr 7 total 5 remain 0 wait 6 TA 11 WTA 2.20
At time 18 process 6 Started arr 7 total 6 remain 6 wait 11
At time 24 process 6 Finshed arr 7 total 6 remain 0 wait 11 TA 17 WTA 2.83
At time 24 process 2 Resumed arr 2 total 4 remain 1 wait 19
At time 25 process 2 Finshed arr 2 total 4 remain 0 wait 19 TA 23 WTA 5.75
At time 25 process 1 Resumed arr 1 total 6 remain 5 wait 23
At time 30 process 1 Finshed arr 1 total 6 remain 0 wait 23 TA 29 WTA 4.83
```

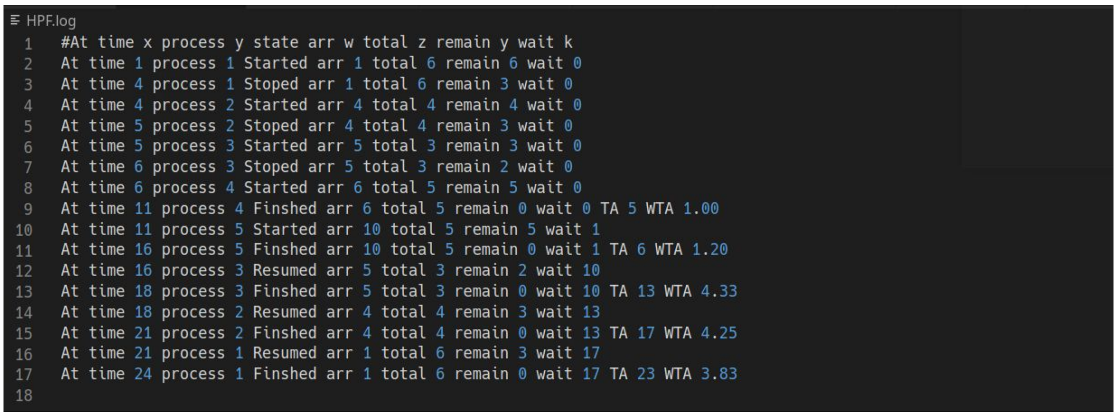


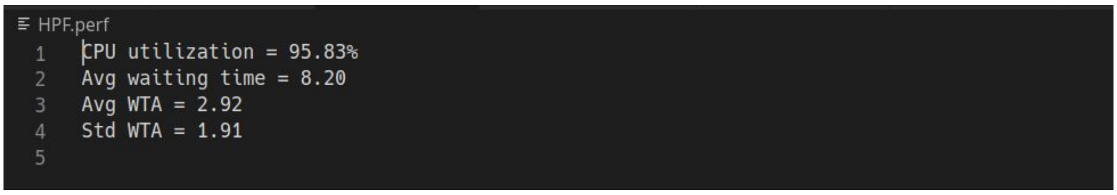
```
CPU utilization = 96.67%
Avg waiting time = 10.67
Avg WTA = 3.21
Std WTA = 1.98
```

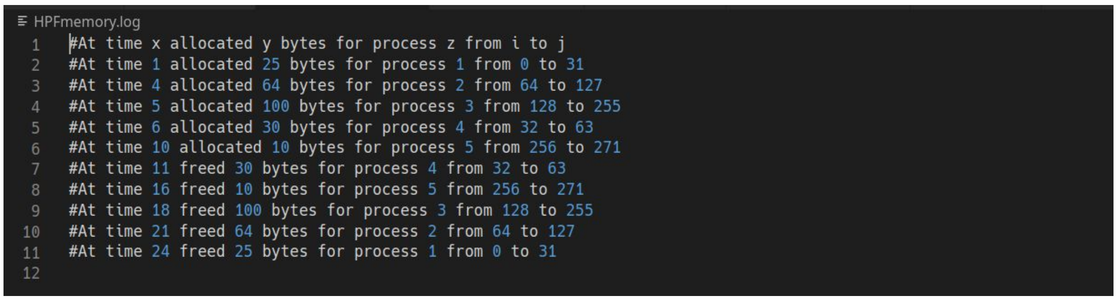

HPFmemory.log
```
#At time x allocated y bytes for process z from i to j
#At time 1 allocated 256 bytes for process 1 from 0 to 255
#At time 2 allocated 64 bytes for process 2 from 256 to 319
#At time 5 allocated 100 bytes for process 3 from 384 to 511
#At time 7 allocated 63 bytes for process 4 from 320 to 383
#At time 7 allocated 120 bytes for process 5 from 512 to 639
#At time 7 allocated 250 bytes for process 6 from 768 to 1023
#At time 12 freed 63 bytes for process 4 from 320 to 383
#At time 13 freed 100 bytes for process 3 from 384 to 511
#At time 18 freed 120 bytes for process 5 from 512 to 639
#At time 24 freed 250 bytes for process 6 from 768 to 1023
#At time 25 freed 64 bytes for process 2 from 256 to 319
#At time 30 freed 256 bytes for process 1 from 0 to 255
```

HPF: test case 2

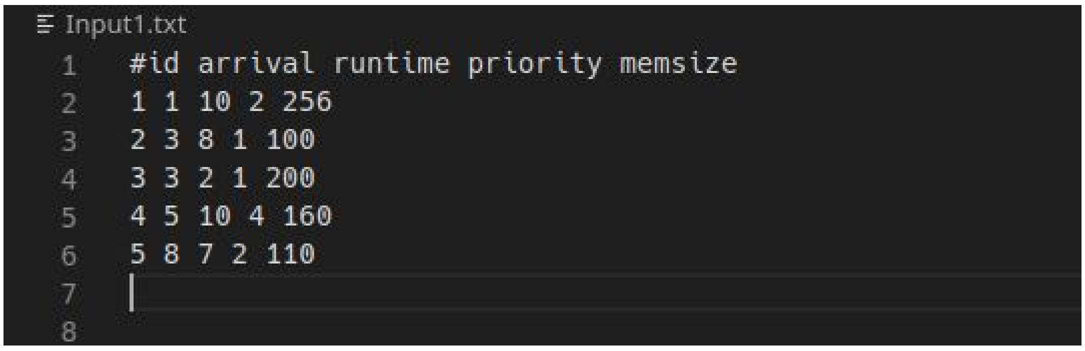

Input1.txt
```
#id arrival runtime priority memsize
1   1   6   5 25
2   4   4   4 64
3   5   3   2 100
4   6   5   1 30
5   10   5  2 10
```


HPF.log
```
#At time x process y state arr w total z remain y wait k
At time 1 process 1 Started arr 1 total 6 remain 6 wait 0
At time 4 process 1 Stoped arr 1 total 6 remain 3 wait 0
At time 4 process 2 Started arr 4 total 4 remain 4 wait 0
At time 5 process 2 Stoped arr 4 total 4 remain 3 wait 0
At time 5 process 3 Started arr 5 total 3 remain 3 wait 0
At time 6 process 3 Stoped arr 5 total 3 remain 2 wait 0
At time 6 process 4 Started arr 6 total 5 remain 5 wait 0
At time 11 process 4 Finshed arr 6 total 5 remain 0 wait 0 TA 5 WTA 1.00
At time 11 process 5 Started arr 10 total 5 remain 5 wait 1
At time 16 process 5 Finshed arr 10 total 5 remain 0 wait 1 TA 6 WTA 1.20
At time 16 process 3 Resumed arr 5 total 3 remain 2 wait 10
At time 18 process 3 Finshed arr 5 total 3 remain 0 wait 10 TA 13 WTA 4.33
At time 18 process 2 Resumed arr 4 total 4 remain 3 wait 13
At time 21 process 2 Finshed arr 4 total 4 remain 0 wait 13 TA 17 WTA 4.25
At time 21 process 1 Resumed arr 1 total 6 remain 3 wait 17
At time 24 process 1 Finshed arr 1 total 6 remain 0 wait 17 TA 23 WTA 3.83
```

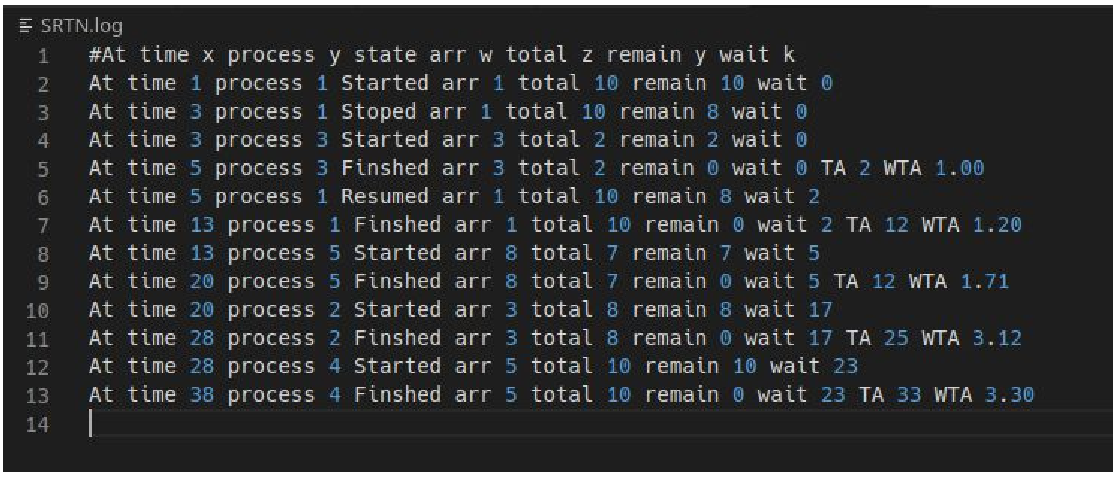

HPF.perf
```
CPU utilization = 95.83%
Avg waiting time = 8.20
Avg WTA = 2.92
Std WTA = 1.91
```


HPFmemory.log
```
#At time x allocated y bytes for process z from i to j
#At time 1 allocated 25 bytes for process 1 from 0 to 31
#At time 4 allocated 64 bytes for process 2 from 64 to 127
#At time 5 allocated 100 bytes for process 3 from 128 to 255
#At time 6 allocated 30 bytes for process 4 from 32 to 63
#At time 10 allocated 10 bytes for process 5 from 256 to 271
#At time 11 freed 30 bytes for process 4 from 32 to 63
#At time 16 freed 10 bytes for process 5 from 256 to 271
#At time 18 freed 100 bytes for process 3 from 128 to 255
#At time 21 freed 64 bytes for process 2 from 64 to 127
#At time 24 freed 25 bytes for process 1 from 0 to 31
```

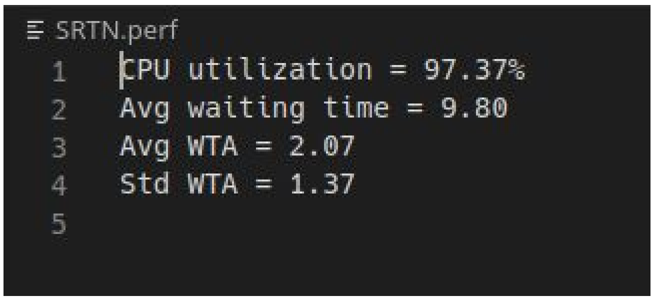
SRTN case 1:


Input1.txt

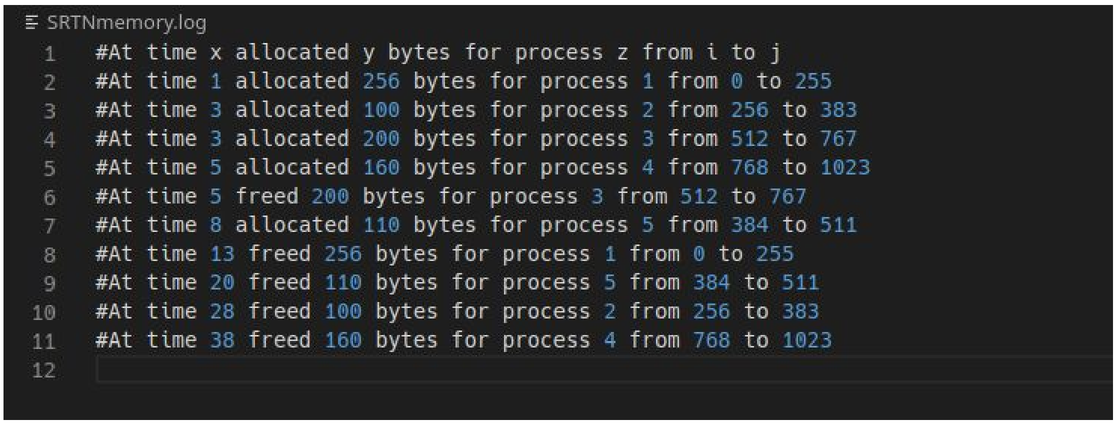
```
#id arrival runtime priority memsize
1 1 10 2 256
2 3 8 1 100
3 3 2 1 200
4 5 10 4 160
5 8 7 2 110
```


SRTN.log

```
#At time x process y state arr w total z remain y wait k
At time 1 process 1 Started arr 1 total 10 remain 10 wait 0
At time 3 process 1 Stoped arr 1 total 10 remain 8 wait 0
At time 3 process 3 Started arr 3 total 2 remain 2 wait 0
At time 5 process 3 Finshed arr 3 total 2 remain 0 wait 0 TA 2 WTA 1.00
At time 5 process 1 Resumed arr 1 total 10 remain 8 wait 2
At time 13 process 1 Finshed arr 1 total 10 remain 0 wait 2 TA 12 WTA 1.20
At time 13 process 5 Started arr 8 total 7 remain 7 wait 5
At time 20 process 5 Finshed arr 8 total 7 remain 0 wait 5 TA 12 WTA 1.71
At time 20 process 2 Started arr 3 total 8 remain 8 wait 17
At time 28 process 2 Finshed arr 3 total 8 remain 0 wait 17 TA 25 WTA 3.12
At time 28 process 4 Started arr 5 total 10 remain 10 wait 23
At time 38 process 4 Finshed arr 5 total 10 remain 0 wait 23 TA 33 WTA 3.30
```

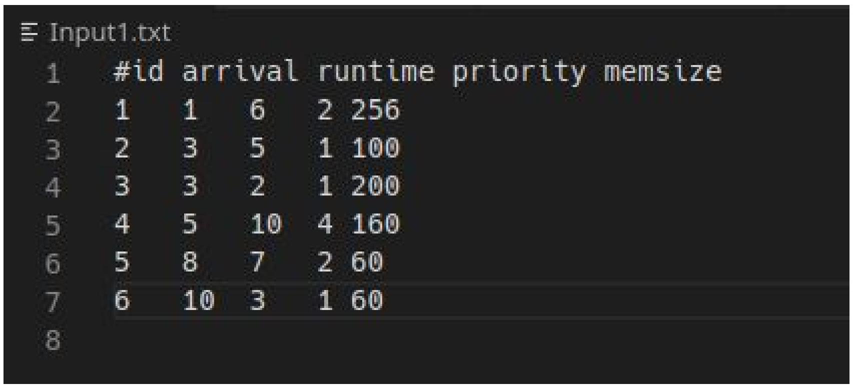

SRTN.perf

```
CPU utilization = 97.37%
Avg waiting time = 9.80
Avg WTA = 2.07
Std WTA = 1.37
```

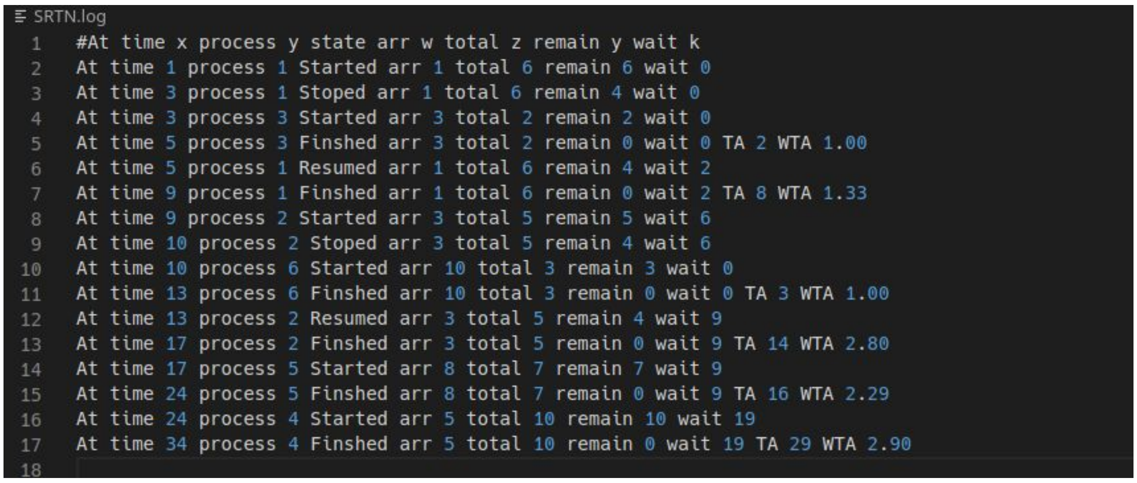

SRTNmemory.log

```
#At time x allocated y bytes for process z from i to j
#At time 1 allocated 256 bytes for process 1 from 0 to 255
#At time 3 allocated 100 bytes for process 2 from 256 to 383
#At time 3 allocated 200 bytes for process 3 from 512 to 767
#At time 5 allocated 160 bytes for process 4 from 768 to 1023
#At time 5 freed 200 bytes for process 3 from 512 to 767
#At time 8 allocated 110 bytes for process 5 from 384 to 511
#At time 13 freed 256 bytes for process 1 from 0 to 255
#At time 20 freed 110 bytes for process 5 from 384 to 511
#At time 28 freed 100 bytes for process 2 from 256 to 383
#At time 38 freed 160 bytes for process 4 from 768 to 1023
```

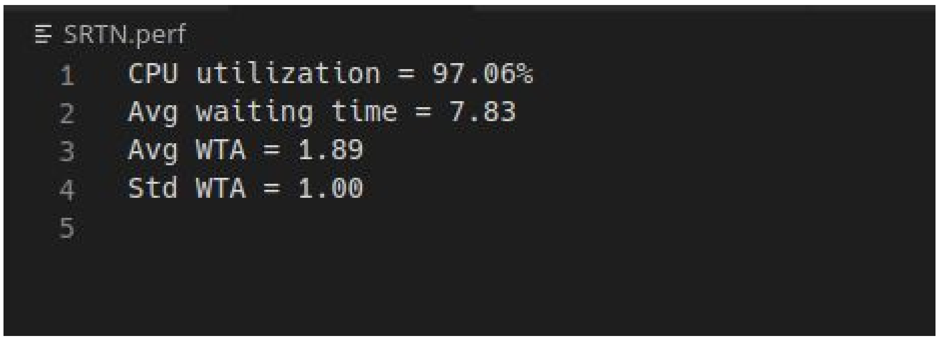
SRTN case 2:


Input1.txt

```
#id arrival runtime priority memsize
1   1   6   2 256
2   3   5   1 100
3   3   2   1 200
4   5   10  4 160
5   8   7   2 60
6   10  3   1 60
```


SRTN.log

```
#At time x process y state arr w total z remain y wait k
At time 1 process 1 Started arr 1 total 6 remain 6 wait 0
At time 3 process 1 Stoped arr 1 total 6 remain 4 wait 0
At time 3 process 3 Started arr 3 total 2 remain 2 wait 0
At time 5 process 3 Finshed arr 3 total 2 remain 0 wait 0 TA 2 WTA 1.00
At time 5 process 1 Resumed arr 1 total 6 remain 4 wait 2
At time 9 process 1 Finshed arr 1 total 6 remain 0 wait 2 TA 8 WTA 1.33
At time 9 process 2 Started arr 3 total 5 remain 5 wait 6
At time 10 process 2 Stoped arr 3 total 5 remain 4 wait 6
At time 10 process 6 Started arr 10 total 3 remain 3 wait 0
At time 13 process 6 Finshed arr 10 total 3 remain 0 wait 0 TA 3 WTA 1.00
At time 13 process 2 Resumed arr 3 total 5 remain 4 wait 9
At time 17 process 2 Finshed arr 3 total 5 remain 0 wait 9 TA 14 WTA 2.80
At time 17 process 5 Started arr 8 total 7 remain 7 wait 9
At time 24 process 5 Finshed arr 8 total 7 remain 0 wait 9 TA 16 WTA 2.29
At time 24 process 4 Started arr 5 total 10 remain 10 wait 19
At time 34 process 4 Finshed arr 5 total 10 remain 0 wait 19 TA 29 WTA 2.90
```


SRTN.perf

```
CPU utilization = 97.06%
Avg waiting time = 7.83
Avg WTA = 1.89
Std WTA = 1.00
```

Work plan and load distribution:

- **A two weeks period (Phase 1):**
  - Starting Thursday, October 27 to Thursday, November 10th.
  - Design the data structure used in the project. (Salma-Radwa)

- **A three-week period (Phase 1):**
  - Starting Thursday, November 10th to Thursday, December 1st.
  - Creating process generator (including the input file) (Salma-Radwa)
  - Implementing the data structure (Queue and Priority Queue) (Salma-Radwa)
  - Initiating the scheduler and process (Salma-Radwa)
  - Creating STF scheduler (Salma-Radwa)

- **A one week period(Phase 1):**
  - Starting Sunday, December 4th to Sunday, December 11th.
  - Creating RR scheduler (Salma)
  - Creating HPF scheduler (Radwa)
  - Creating the output files (.log and .perf) (Radwa)
  - Clear IPC resources in the process generator. (Salma)

- **A one week period (Phase 2):**
  - Starting Friday, December 25th to Thursday, December 31st.
  - Design the new data structure (including binary tree and the leaves queue) (Salma-Radwa)
  - Creating the Allocation function in SJF scheduler. (Radwa)
  - Creating the deallocation function in SJF scheduler. (Salma)

- **A two week period (Phase 2):**
    - Starting Sunday, January 10th to Tuesday, January 25th.
    - Creating SRTN scheduler. (Salma)
    - Applying the allocation function in all schedulers. (Salma)
    - Applying the deallocation function in all schedulers. (Radwa)
    - Creating the memory output file for all schedulers. (Radwa)

References:

[1] www.rosettacode.org/wiki/Rosetta_Code

[2]https://www.geeksforgeeks.org/priority-queue-using-linked-list/#:~:text=Priority%20Queues%20can%20be%20implemented,lists%2C%20heaps%20and%20binary%20trees.&text=The%20list%20is%20so%20created,elements%20based%20on%20their%20priority.