A
Project Report
On
IMPLEMENTATION OF CPU SCHEDULING ALGORITHMS IN C++ WITH ENCRYPTED OUTPUTS

- *Objective*

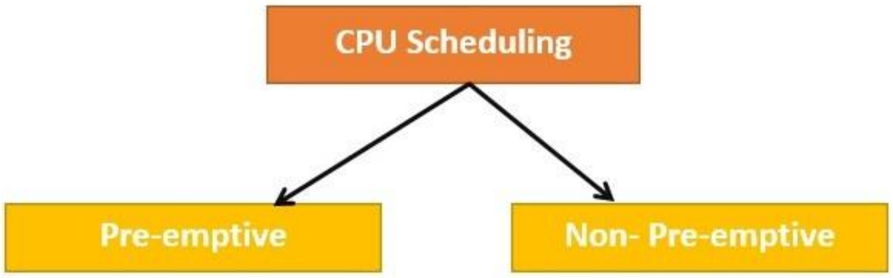
To schedule the operating system processes based on the CPU-Scheduling Algorithms.
The objective is to develop a model CPU scheduler to guard against runway dispatch latency, and guarantee an optimum level of scheduler efficiency.

- RSN Sasmith Reddy (20JE0740)

- ***INTRODUCTION***

CPU SCHEDULING :

In a single-processor system, only one process can run at a time; any others must wait until the CPU is free and can be rescheduled. The objective of multiprogramming is to have some process running at all times, to maximise CPU utilisation. Scheduling is a fundamental operating-system function. Almost all computer resources are scheduled before use. The CPU is, of course, one of the primary computer resources. Thus, its scheduling is central to operating-system design. CPU scheduling determines which processes run when there are multiple run-able processes.



- ***Preemptive Scheduling***

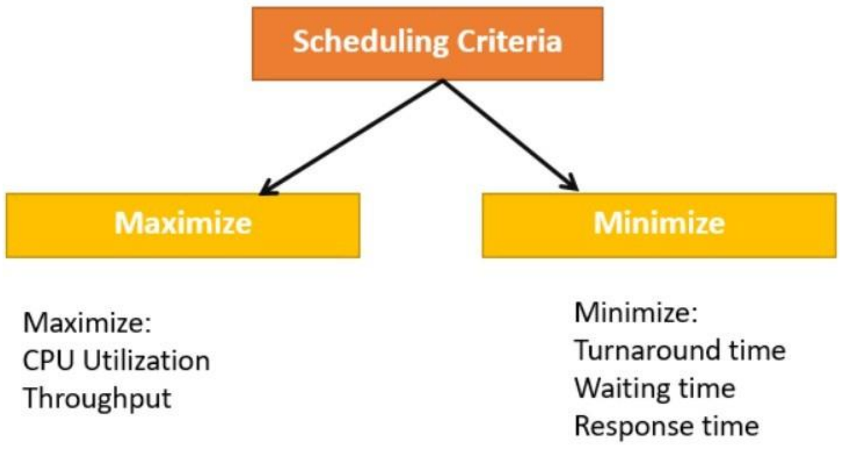
In Preemptive Scheduling, the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running. The lower priority task holds for some time and resumes when the higher priority task finishes its execution.

- ***Non-Preemptive Scheduling***

In this type of scheduling method, the CPU has been allocated to a specific process. The process that keeps the CPU busy will release the CPU either by switching context or terminating. It is the only method that can be used for various hardware platforms. That’s because it doesn’t need special hardware (for example, a timer) like preemptive scheduling.

The ***Scheduler*** is concerned mainly with :

- ***CPU Utilisation*** **:** time utilised by CPU for a process.
- ***Throughput*** **:** The number of processes executed in a specified time period is called its throughput.
- ***Turnaround Time*** **:** The amount of time that is needed to execute a process is called turnaround time. It is the actual job time plus the waiting time.
- ***Waiting Time*** **:** The amount of time the process has waited is called waiting time. It is the turnaround time minus actual **job** time.
- ***Response Time*** **:** The amount of time between a request is Submitted and the first response being produced is called response time.



Different algorithms are used for CPU scheduling . Such as

- **First Come First Serve**
- **Shortest Job First**
- **Shortest Remaining Time First**
- **Priority Based (Preemptive and Non-preemptive)**
- **Round Robin**
- **Longest Job First**
- **Longest Remaining Time First**
- **Highest Response Ratio Next**
- **Multilevel Feedback Queue**
- **Multilevel Queue**

- ***<u>METHODOLOGY</u>***

Our project includes implementation of below algorithms in C++ :

1.) FIRST-COME- FIRST-SERVED

- With this scheme, whichever process requests CPU time first is allocated the CPU first. This is easily implemented with a **<u>FIFO queue</u>** for managing the tasks; as they come in, they're put on the end of the queue. As the CPU finishes each task, it pops it off the start of the queue and heads on to the next one.

2.) SHORTEST JOB FIRST

- CPU is then given to the process with the minimal CPU burst from the waiting queue. SJF is provably optimal, in that for a given set of processes and their CPU bursts/execution times it gives the least average waiting time for each process.
- SJF algorithm may be ***preemptive*** or ***non- preemptive*** . A preemptive SJF algorithm will preempt the currently executing process, whereas a non-preemptive SJF algorithm will allow the currently running process to finish its CPU burst. Preemptive SJF scheduling is sometimes called ***shortest-remaining-time-first***.

3.) SHORTEST REMAINING TIME

- It is the ***preemptive*** version of shortest job first algo A preemptive SJF algorithm will preempt the currently executing process it is done with preemption of process comparatively long if any process comes with shortest burst time.

4.) PRIORITY BASED (PREEMPTIVE AND NON-PREEMPTIVE)

- In Priority scheduling, there is a priority number assigned to each process. In some systems, the lower the number, the higher the priority. While, in the others, the higher the number, the higher will be the priority. The Process with the ***higher priority*** among the available processes is given the CPU.
- ***<u>Preemptive Priority Scheduling</u>*** **:** If the new process arriving at the ready queue has a higher priority than the currently running process, the CPU is preempted, which means the processing of the current process is stopped and the incoming new process with higher priority gets the CPU for its execution.
- **<u>Non-Preemptive Priority Scheduling</u>** ***:*** In case of a non-preemptive priority scheduling algorithm if a new process arrives with a higher priority than the current running process, the incoming process is put at the head of the ready queue, which means after the execution of the current process it will be processed.

5.) ROUND ROBIN

- Round Robin scheduling algorithm is one of the most popular scheduling algorithms .
- The Algorithm focuses on ***Time Sharing***. In this algorithm, every process gets executed in a cyclic way. A certain ***time slice*** is defined in the system which is called time quantum. Each process present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time then the process will terminate else the process will go back to the ready queue and wait for the next turn to complete the execution.

6.) LONGEST JOB FIRST (PREEMPTIVE AND NON-PREEMPTIVE)

- Out of all the available processes, the CPU is assigned to the process having the largest burst time.
- In case of a tie, it is broken by ***FCFS*** Scheduling.
- ***LJF Scheduling*** can be used in both ***preemptive*** and ***non-preemptive mode.***
- Preemptive mode of Longest Job First is called ***Longest Remaining Time First (LRTF)***.

7.) HIGHEST RESPONSE RATIO NEXT

- ***Highest Response Ratio Next (HRNN)*** is one of the most optimal scheduling algorithms. This is a ***non-preemptive algorithm*** in which the scheduling is done on the basis of an extra parameter called ***Response Ratio***. A Response Ratio is calculated for each of the available jobs and the Job with the highest response ratio is given priority over the others.
- **Response Ratio = (W+S)/S**

  {

  W = It indicates the **Waiting Time.**

  S = It indicates the Service time that is **Burst Time**.

  }

8.) MULTILEVEL QUEUE

- Multilevel queue scheduling is used when processes in the ready queue can be divided into ***different classes*** where each class has its own scheduling needs.
- For instance, ***foreground*** or ***interactive*** processes and ***background*** or ***batch*** processes are commonly divided. Foreground and background processes have different time requirements and hence will have different scheduling needs. In this case, multilevel queue scheduling will be used.
- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system. Processes do not move between queues. This setup has the advantage of low scheduling overhead, but the disadvantage of being inflexible.

9.) MULTILEVEL FEEDBACK QUEUE

- Multilevel feedback queue scheduling, however, allows a process to move between queues. The idea is to separate processes with ***different CPU-burst characteristics***. If a process uses too much CPU time, it will be moved to a lower-priority queue. Similarly, a process that waits too long in a lower-priority queue may be moved to a higher-priority queue. This form of ageing prevents starvation.
- Multilevel-feedback-queue scheduler defined by the following parameters :
  - number of queues scheduling algorithms for each queue.
  - method used to determine when to upgrade a process.
  - method used to determine when to demote a process.
  - method used to determine which queue a process will enter when that process needs service.

Overview

Scheduling Algorithm	Advantages	Disadvantages
FCFS	- Simple and easy to implement	- Waiting time is high - Lower CPU utilisation
SJF	- Minimum average waiting time	- Causes starvation
SRTF	- Faster than SJF	- Increased context switching
RR	- No starvation	- Average waiting time is high
Priority based	- Suitable for applications with fluctuating time and resource requirements	- Lower priority processes may starve
HRRN	- Limits waiting time of longer jobs and also supports shorter jobs	- Can't be implemented practically
LJF	- All processes approximately finishes at same time	- Reduction in the utilisation of CPU - Increase average turn around time
Multi-level queue	- Separate scheduling for various kinds of processes is possible	- Faces starvation
Multi-level feedback queue	- Low scheduling overhead	- Most complex and not flexible

Techniques used to encrypt output file :

1) Key-Based Encryption

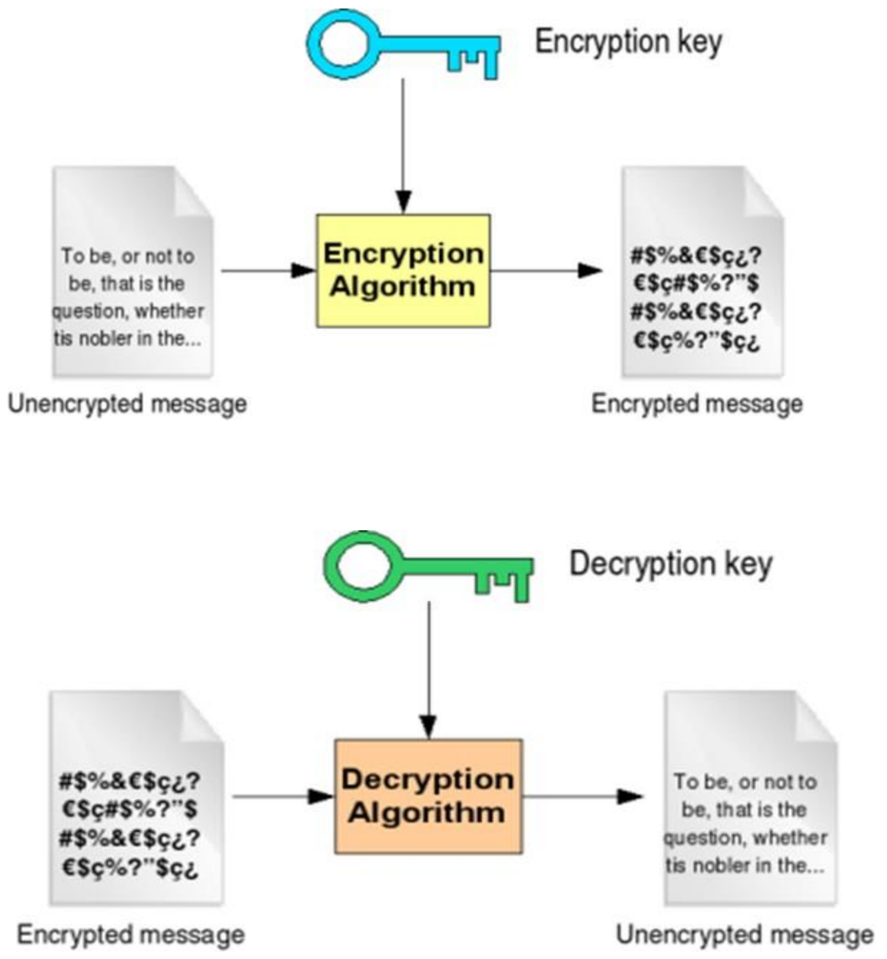
2) MD5 Based Encryption

- Key-Based Encryption :

Symmetric-Key Encryption :

In this type of cryptographic Technique, only one key is used to encrypt and decrypt the message or the file.

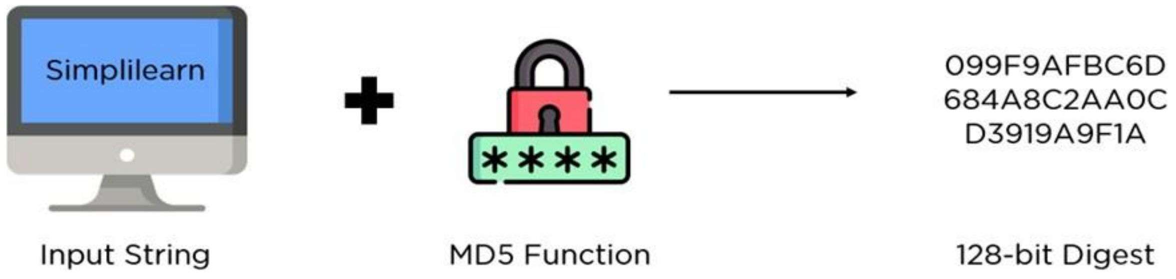
The original data is known as the **plaintext**, and the data after the key encrypts it is known as the **Ciphertext.**


Encryption key
To be, or not to be, that is the question, whether tis nobler in the...
Unencrypted message
Encryption Algorithm
#$%&€$ç¿? €$ç#$%?"$ #$%&€$ç¿? €$ç%?"$ç¿
Encrypted message
Decryption key
#$%&€$ç¿? €$ç#$%?"$ #$%&€$ç¿? €$ç%?"$ç¿
Encrypted message
Decryption Algorithm
To be, or not to be, that is the question, whether tis nobler in the...
Unencrypted message

Here we have used the rand() function to generate a random key which is used to encrypt and decrypt the message.

- ***MD5 Based Encryption*** :

MD5 (Message Digest Method 5) is a cryptographic hash algorithm used to generate a 128-bit digest from a string of any length. It represents the digests as 32 digit hexadecimal numbers.



The digest size is always 128 bits and a minor change in the input string generates a drastically different digest .This is essential to prevent similar Hash generation as much as possible.

There are Four Major Steps in This Encryption Algorithm :

1) Padding bits

2) Padding Length

3) Initialise MD Buffer

4) Process Each Block

Advantages Of MD5 :

Easy to Compare **:**

Unlike the latest hash algorithm families, a 32 digit digest is relatively easier to compare when verifying the digests.

Storing Passwords **:**

Passwords need not be stored in plaintext format, making them accessible for hackers and malicious actors. When using digests, the database also gets a boost since the size of all hash values will be the same.

Low Resource **:**

A relatively low memory footprint is necessary to integrate multiple services into the same framework without a CPU overhead.

Integrity Check **:**

You can monitor file corruption by comparing hash values before and after transit. Once the hashes match, file integrity checks are valid, and it avoids data corruption.

- *Acronyms and Abbreviations Used :*

PID	- Process ID
AT	- Arrival Time
BT	- Burst Time
CT	- Completion Time
TAT	- Turnaround Time
WT	- Waiting Time
RT	- Remaining Burst Time
Priority	- Priority of Process
avg_WT	- Average Waiting Time
avg_TAT	- Average Turnaround Time

- *User Characteristics :*

User selects the CPU-Scheduling Algorithm and gives the data input in the format of the text file.
User selects the type of output encryption format.

- *Input File Requirements :*

Number of Processes - (For all Algorithms)
Arrival Times - (For all Algorithms)
Burst Times - (For all Algorithms)
Time Quantum - (For Round Robin and MultiLevel Feedback Queue Algorithms)
Priority Number - (For Priority-based Algorithm)
Queue Number - (For MultiLevel Queue Algorithm)

- *System Requirements :*

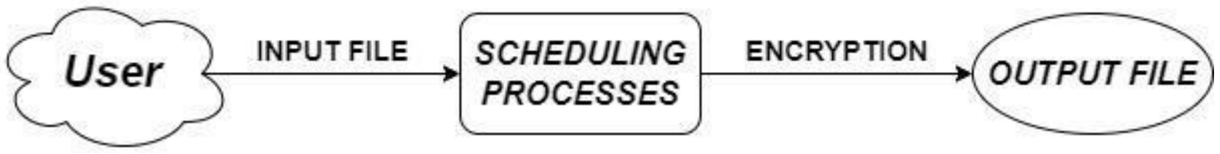
The system shall be compatible to work as a command window based application in C++ .
The system shall meet all performance requirements running on the C++ application .

- *Decrypted Output File Contents :*

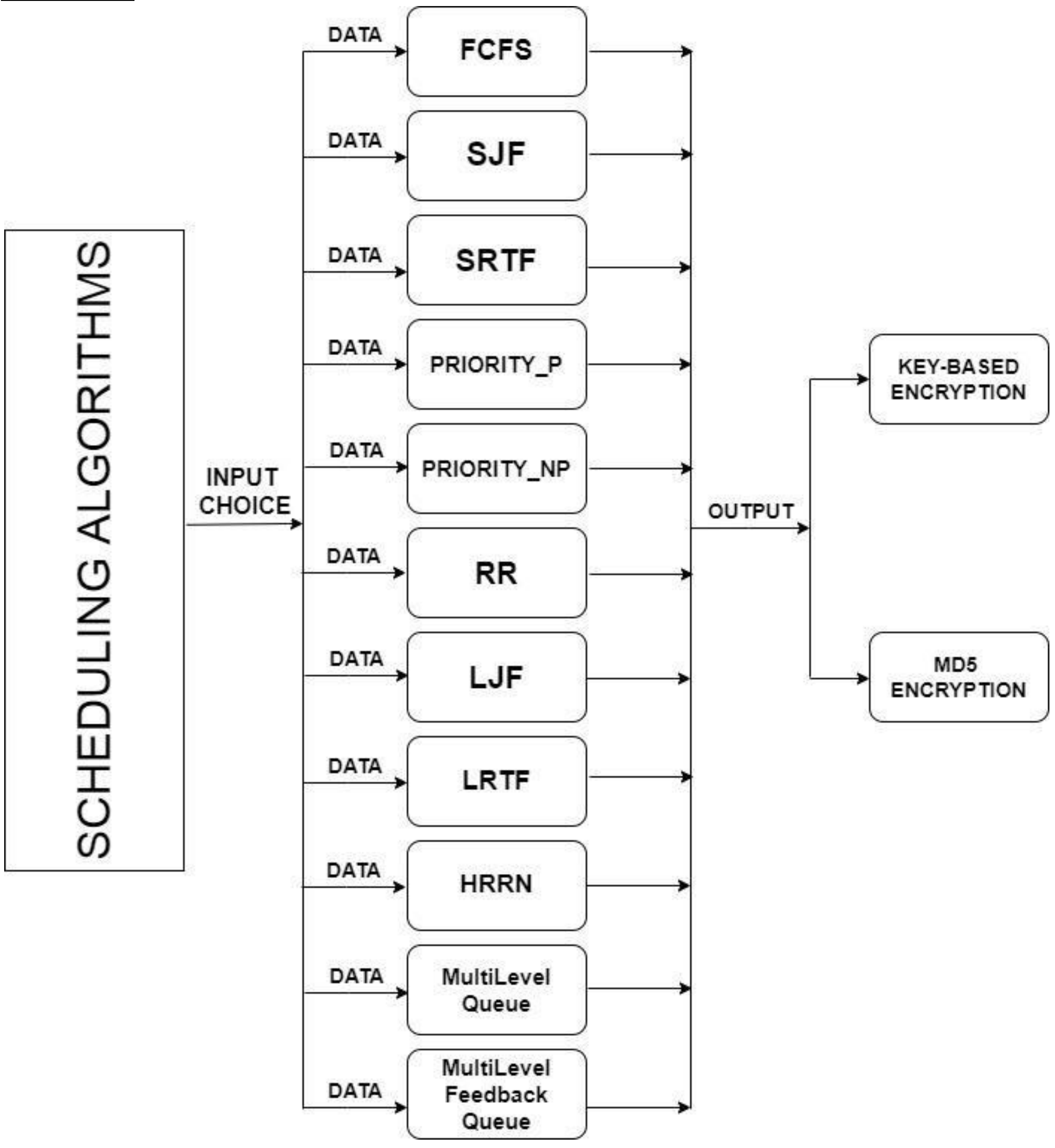
Process Table.
Total and Average Turnaround and Waiting Times.

FlowChart

Level-0 :

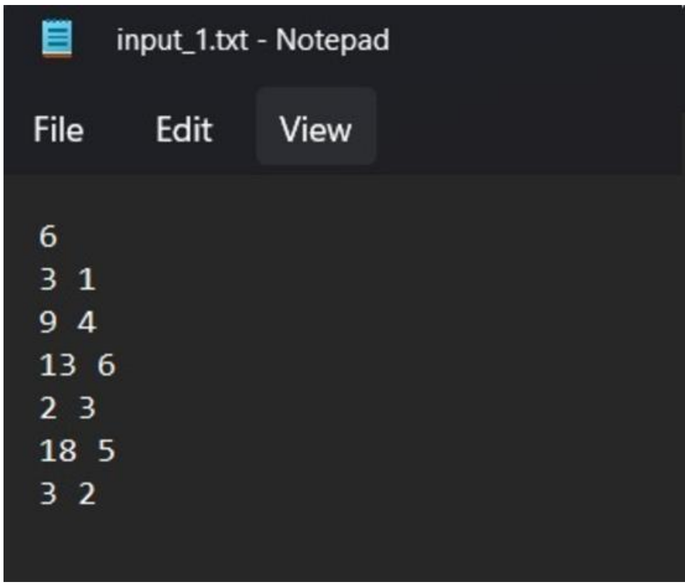


Level-1 :

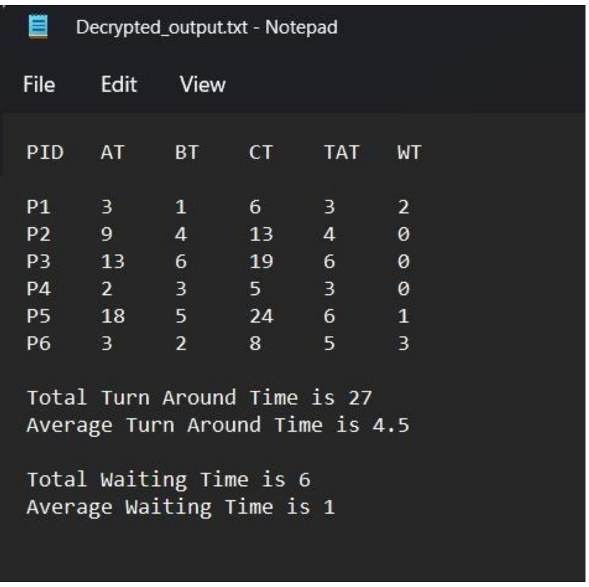


- ***<u>RESULTS</u>* (OUTPUT SNIPPETS)**

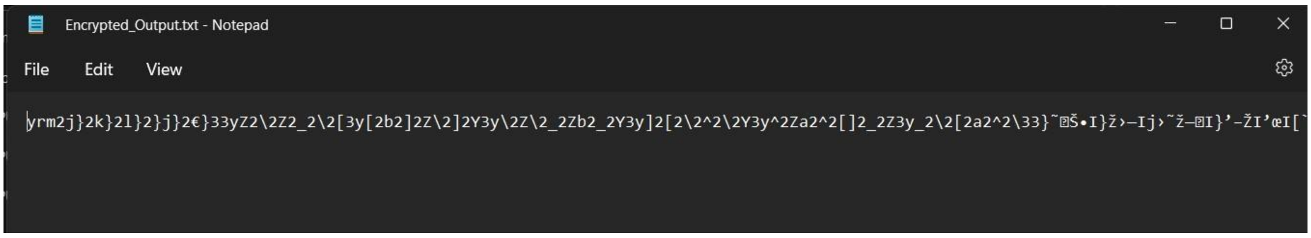
- **<u>FCFS</u> :**



```
6
3 1
9 4
13 6
2 3
18 5
3 2
```

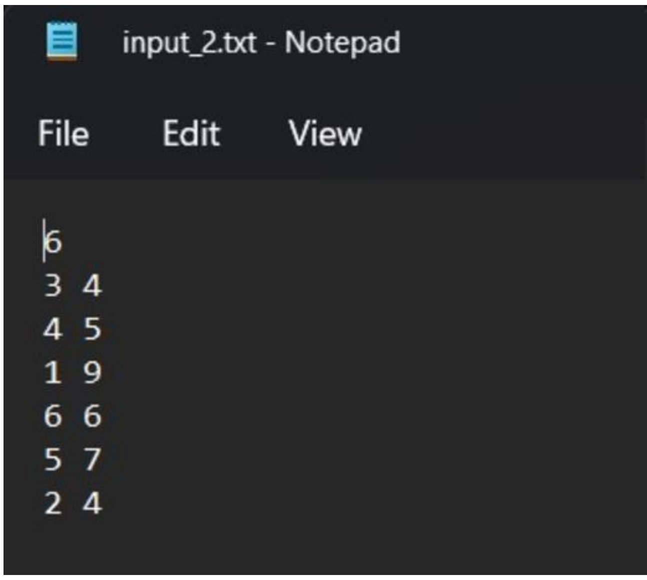


```
PID   AT    BT    CT    TAT   WT

P1    3     1     6     3     2
P2    9     4     13    4     0
P3    13    6     19    6     0
P4    2     3     5     3     0
P5    18    5     24    6     1
P6    3     2     8     5     3

Total Turn Around Time is 27
Average Turn Around Time is 4.5

Total Waiting Time is 6
Average Waiting Time is 1
```

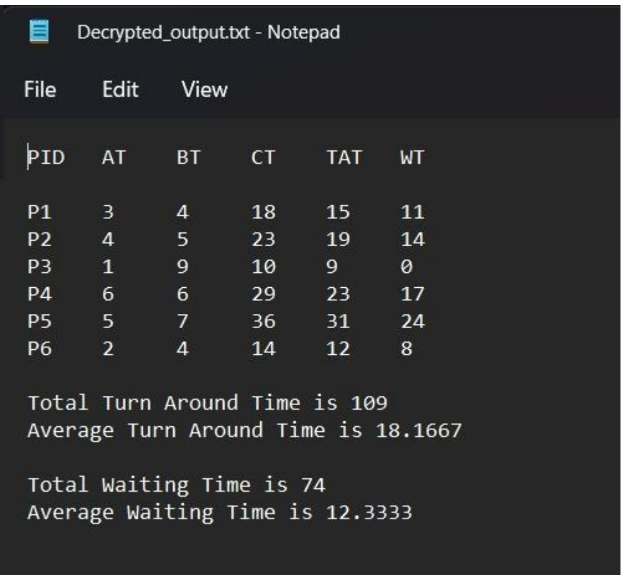


```
yrm2j}2k}2l}2}j}2€}33yZ2\2Z2_2\2[3y[2b2]2Z\2]2Y3y\2Z\2_2Zb2_2Y3y]2[2\2^2\2Y3y^2Za2^2[]2_2Z3y_2\2[2a2^2\33}˜⍰Š•I}ž›–Ij›˜ž–⍰I}’–ŽI’œI[`
```

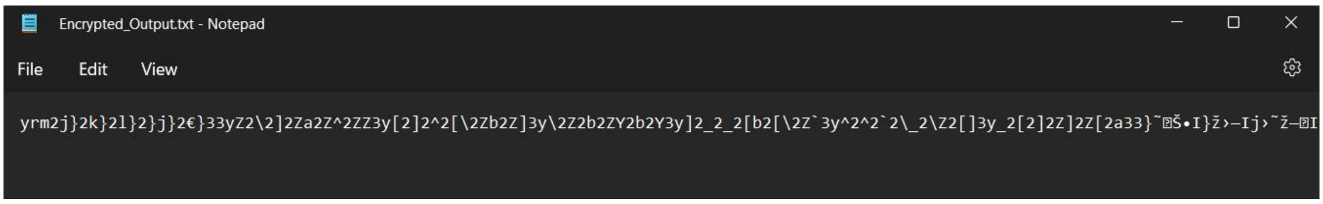
- **<u>SJF</u> :**



```
6
3 4
4 5
1 9
6 6
5 7
2 4
```

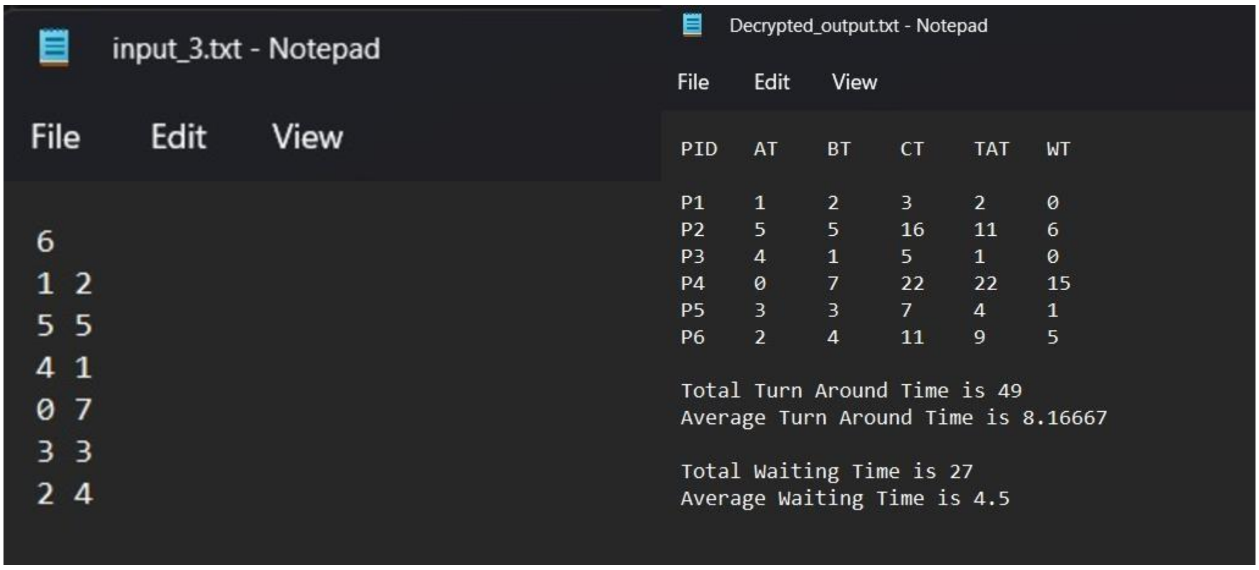


```
PID   AT    BT    CT    TAT   WT

P1    3     4     18    15    11
P2    4     5     23    19    14
P3    1     9     10    9     0
P4    6     6     29    23    17
P5    5     7     36    31    24
P6    2     4     14    12    8

Total Turn Around Time is 109
Average Turn Around Time is 18.1667

Total Waiting Time is 74
Average Waiting Time is 12.3333
```



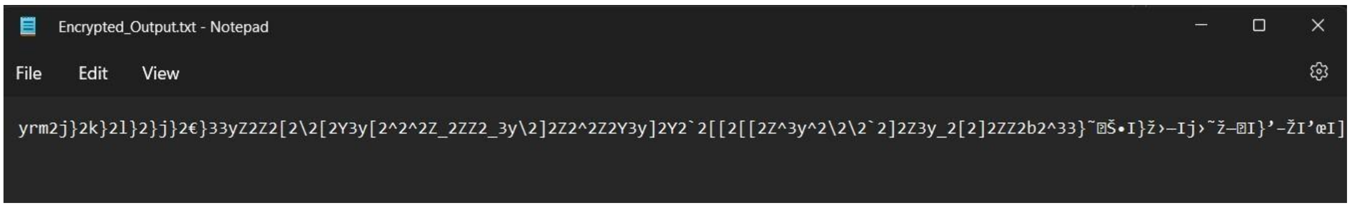
```
yrm2j}2k}2l}2}j}2€}33yZ2\2]2Za2Z^2ZZ3y[2]2^2[\2Zb2Z]3y\2Z2b2ZY2b2Y3y]2_2_2[b2[\2Z`3y^2^2`2\_2\Z2[]3y_2[2]2Z]2Z[2a33}˜⍰Š•I}ž›–Ij›˜ž–⍰I
```

- **SRTF:**



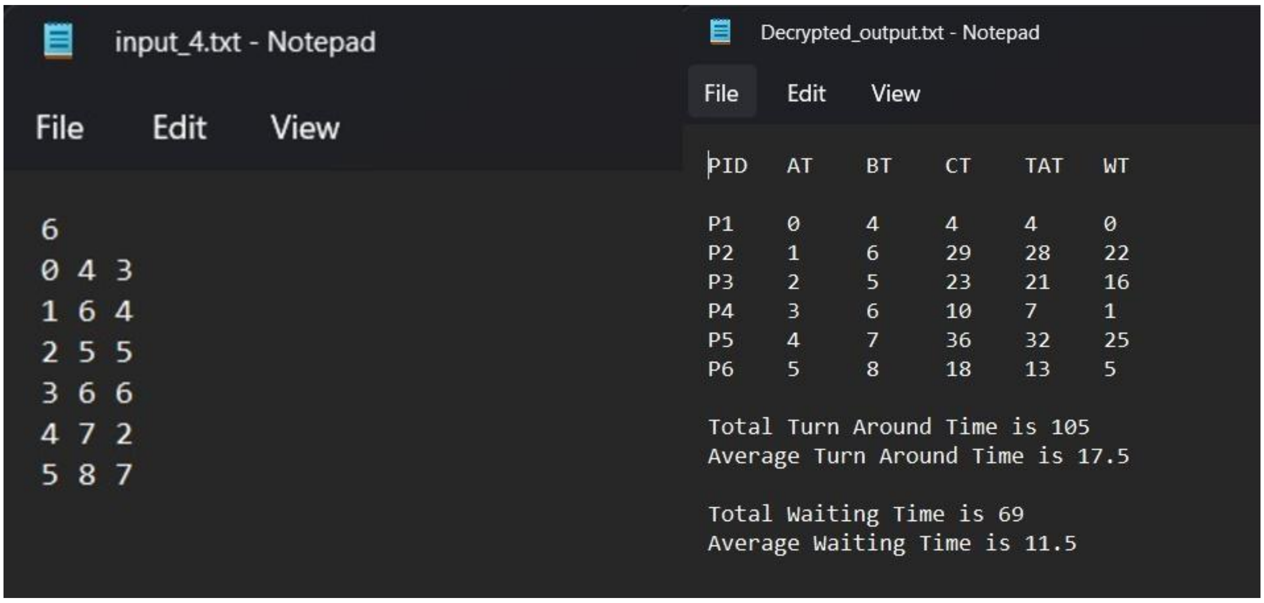
```
6
1 2
5 5
4 1
0 7
3 3
2 4
```

```
PID   AT    BT    CT    TAT   WT

P1    1     2     3     2     0
P2    5     5     16    11    6
P3    4     1     5     1     0
P4    0     7     22    22    15
P5    3     3     7     4     1
P6    2     4     11    9     5

Total Turn Around Time is 49
Average Turn Around Time is 8.16667

Total Waiting Time is 27
Average Waiting Time is 4.5
```



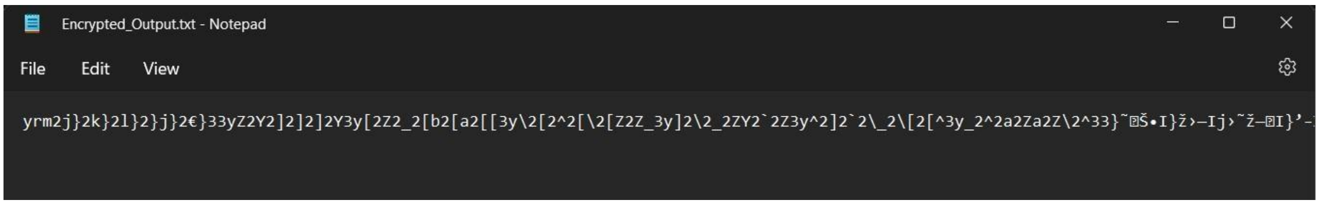
```
yrm2j}2k}2l}2}j}2€}33yZ2Z2[2\2[2Y3y[2^2^2Z_2ZZ2_3y\2]2Z2^2Z2Y3y]2Y2`2[[2[[2Z^3y^2\2\2`2]2Z3y_2[2]2ZZ2b2^33}˜Š•I}ž›–Ij›˜ž–I}’–ŽI’œI]
```

- **PRIORITY(NON-PREEMPTIVE) :**



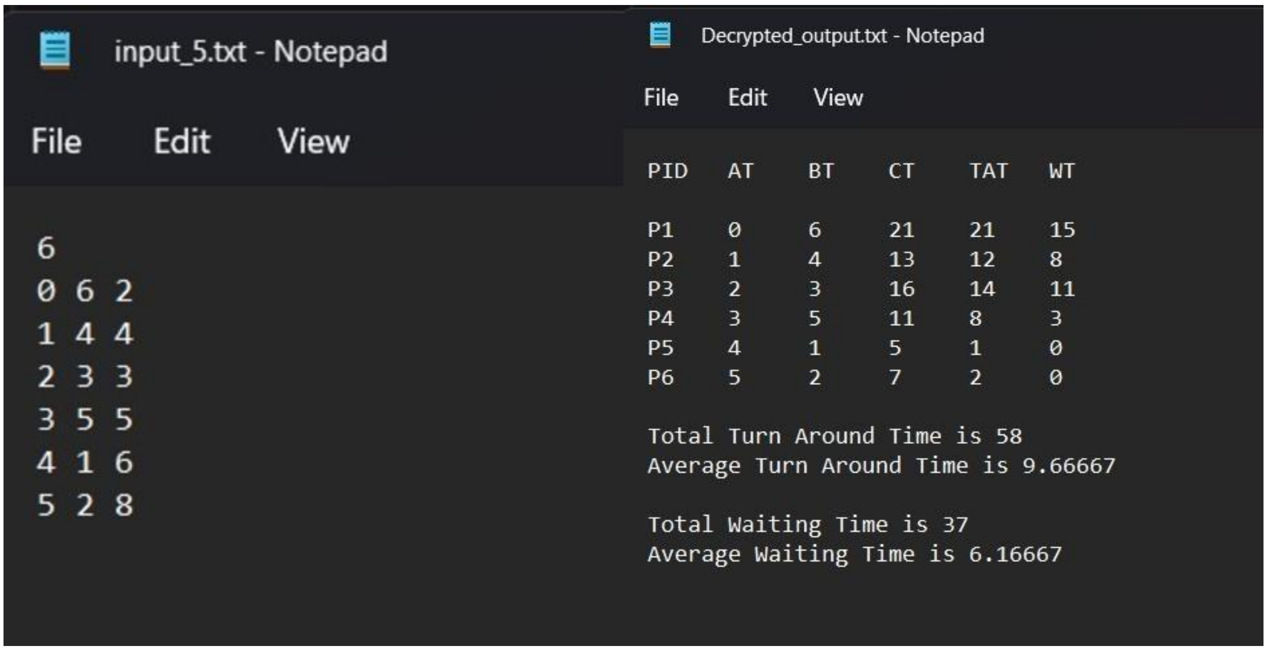
```
6
0 4 3
1 6 4
2 5 5
3 6 6
4 7 2
5 8 7
```

```
PID   AT    BT    CT    TAT   WT

P1    0     4     4     4     0
P2    1     6     29    28    22
P3    2     5     23    21    16
P4    3     6     10    7     1
P5    4     7     36    32    25
P6    5     8     18    13    5

Total Turn Around Time is 105
Average Turn Around Time is 17.5

Total Waiting Time is 69
Average Waiting Time is 11.5
```



```
yrm2j}2k}2l}2}j}2€}33yZ2Y2]2]2]2Y3y[2Z2_2[b2[a2[[3y\2[2^2[\2[Z2Z_3y]2\2_2ZY2`2Z3y^2]2`2\_2\[2[^3y_2^2a2Za2Z\2^33}˜Š•I}ž›–Ij›˜ž–I}’–
```

- **<u>PRIORITY (PRE-EMPTIVE)</u> :**

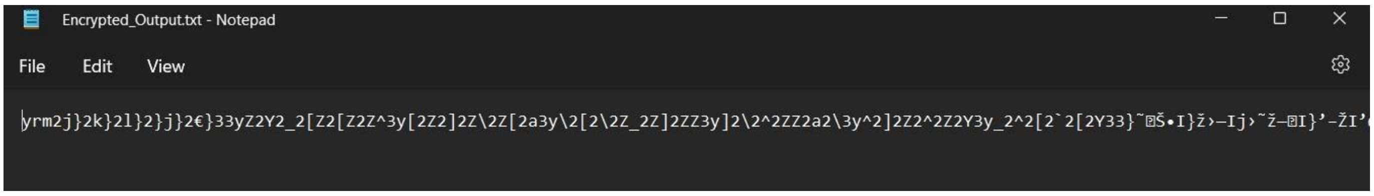


input_5.txt - Notepad

```
6
0 6 2
1 4 4
2 3 3
3 5 5
4 1 6
5 2 8
```

Decrypted_output.txt - Notepad

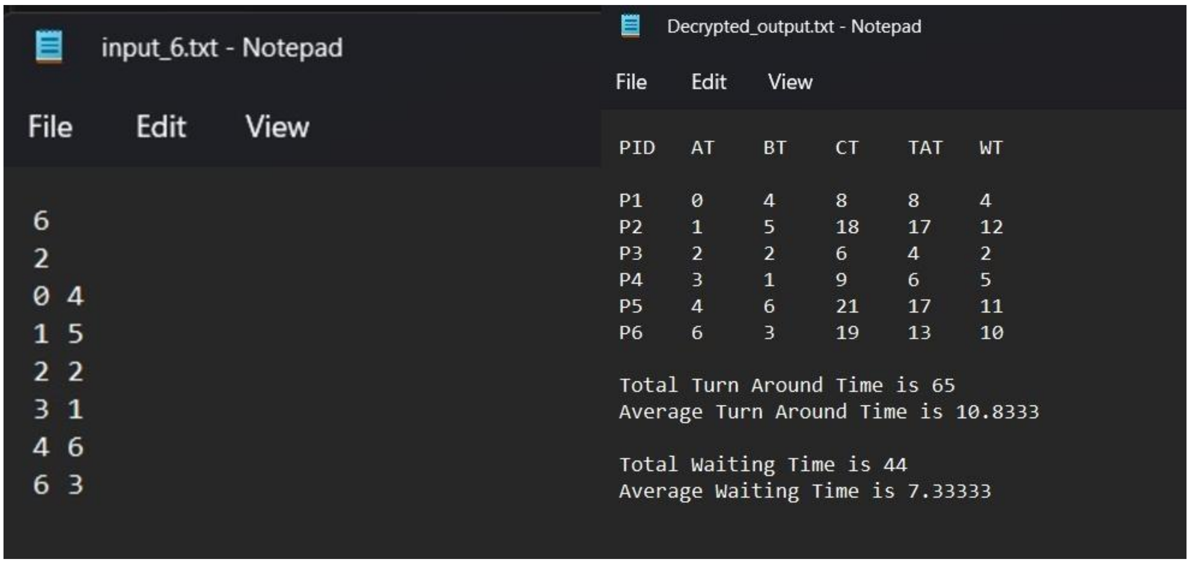
```
PID   AT    BT    CT    TAT   WT

P1    0     6     21    21    15
P2    1     4     13    12    8
P3    2     3     16    14    11
P4    3     5     11    8     3
P5    4     1     5     1     0
P6    5     2     7     2     0

Total Turn Around Time is 58
Average Turn Around Time is 9.66667

Total Waiting Time is 37
Average Waiting Time is 6.16667
```



Encrypted_Output.txt - Notepad

```
yrm2j}2k}2l}2}j}2€}33yZ2Y2_2[Z2[Z2Z^3y[2Z2]2Z\2Z[2a3y\2[2\2Z_2Z]2ZZ3y]2\2^2ZZ2a2\3y^2]2Z2^2Z2Y3y_2^2[2`2[2Y33}˜Š•I}ž›–Ij›˜ž–I}’–ŽI’
```

- **<u>ROUND ROBIN</u> :**

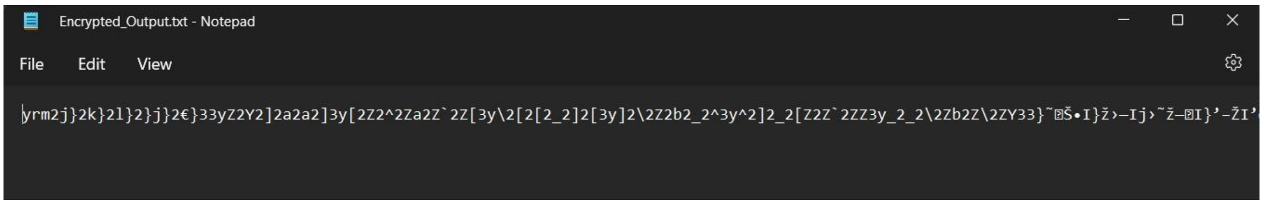


input_6.txt - Notepad

```
6
2
0 4
1 5
2 2
3 1
4 6
6 3
```

Decrypted_output.txt - Notepad

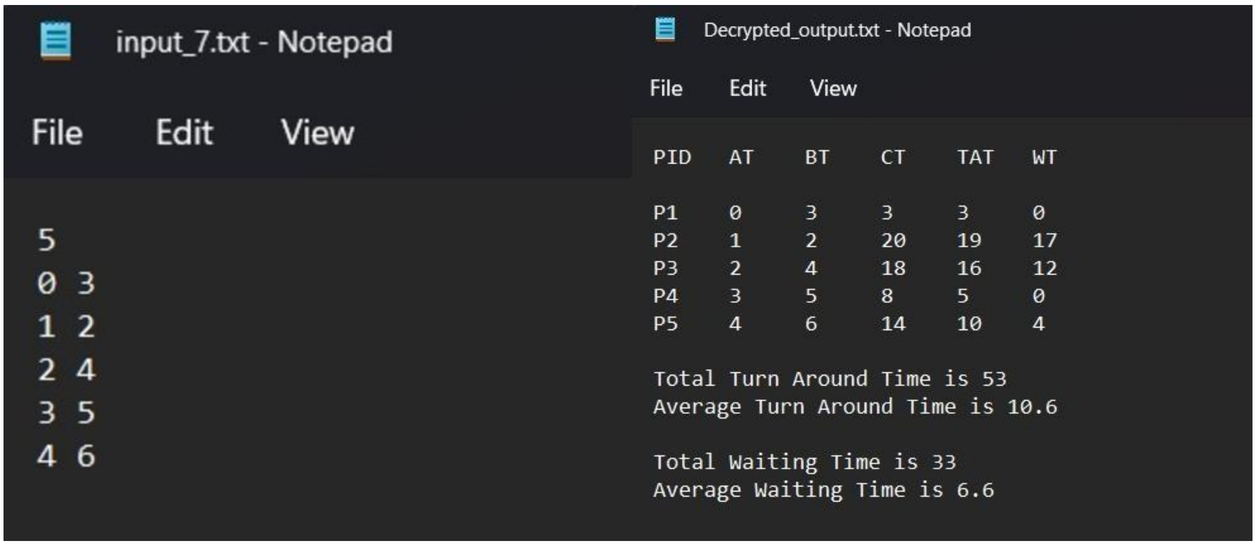
```
PID   AT    BT    CT    TAT   WT

P1    0     4     8     8     4
P2    1     5     18    17    12
P3    2     2     6     4     2
P4    3     1     9     6     5
P5    4     6     21    17    11
P6    6     3     19    13    10

Total Turn Around Time is 65
Average Turn Around Time is 10.8333

Total Waiting Time is 44
Average Waiting Time is 7.33333
```



Encrypted_Output.txt - Notepad

```
yrm2j}2k}2l}2}j}2€}33yZ2Y2]2a2a2]3y[2Z2^2Za2Z`2Z[3y\2[2[2_2]2[3y]2\2Z2b2_2^3y^2]2_2[Z2Z`2ZZ3y_2_2\2Zb2Z\2ZY33}˜Š•I}ž›–Ij›˜ž–I}’–ŽI’
```

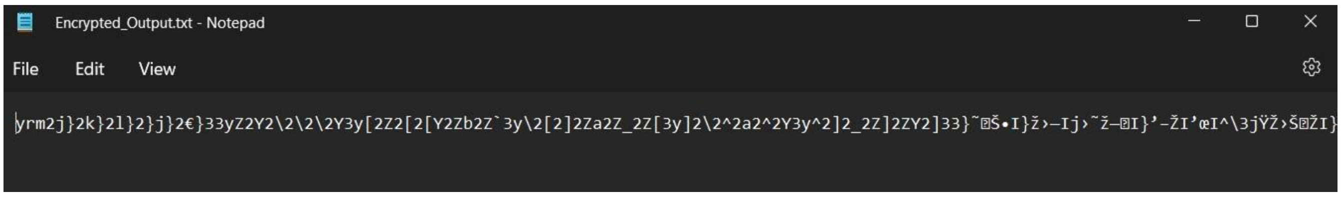
- **<u>LJF</u> :**


input_7.txt - Notepad

```
5
0 3
1 2
2 4
3 5
4 6
```

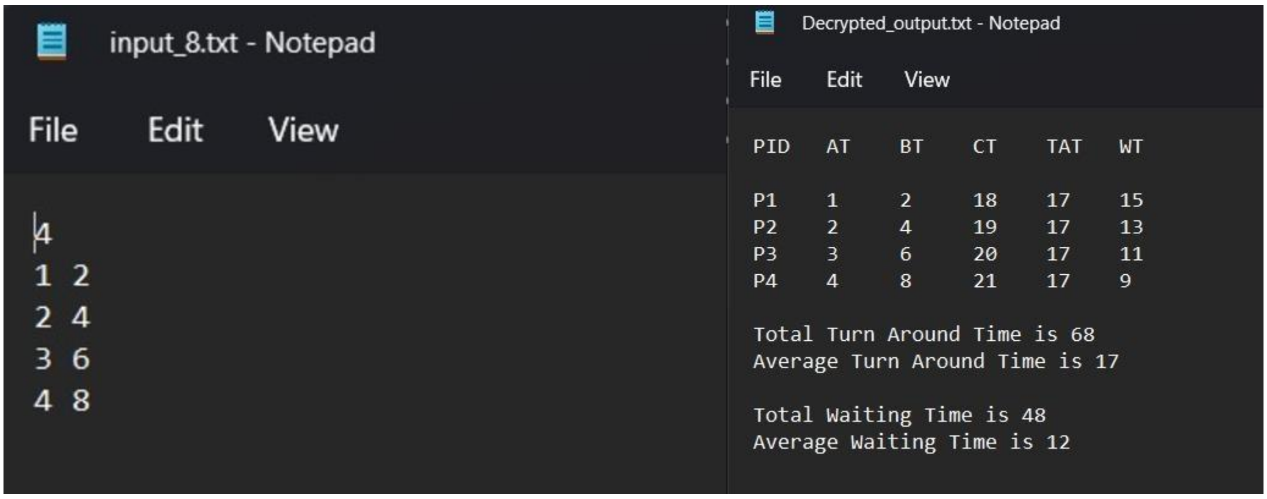
Decrypted_output.txt - Notepad

```
PID   AT    BT    CT    TAT   WT

P1    0     3     3     3     0
P2    1     2     20    19    17
P3    2     4     18    16    12
P4    3     5     8     5     0
P5    4     6     14    10    4

Total Turn Around Time is 53
Average Turn Around Time is 10.6

Total Waiting Time is 33
Average Waiting Time is 6.6
```


Encrypted_Output.txt - Notepad

```
yrm2j}2k}2l}2}j}2€}33yZ2Y2\2\2\2Y3y[2Z2[2[Y2Zb2Z`3y\2[2]2Za2Z_2Z[3y]2\2^2a2^2Y3y^2]2_2Z]2ZY2]33}˜▯Š•I}ž›–Ij›˜ž–▯I}’–ŽI’œI^\3jŸŽ›Š▯ŽI}
```

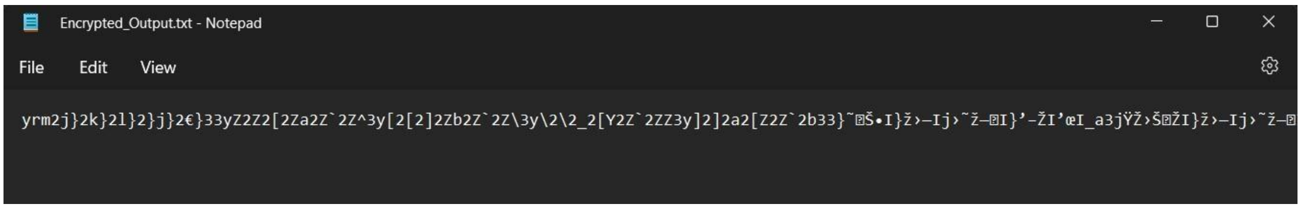
- **<u>LRTF</u> :**


input_8.txt - Notepad

```
4
1 2
2 4
3 6
4 8
```

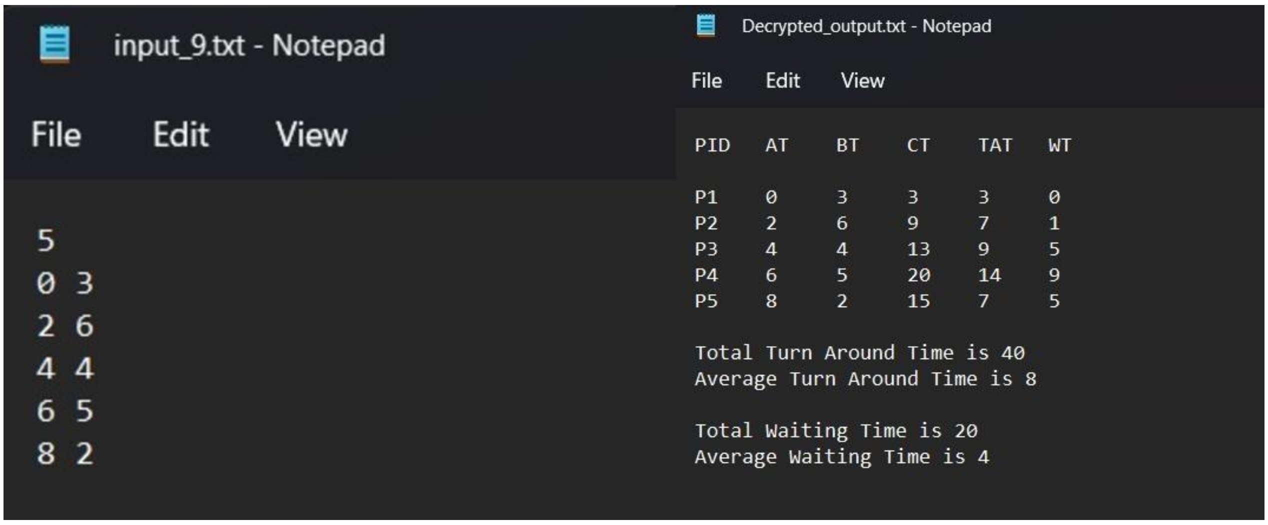
Decrypted_output.txt - Notepad

```
PID   AT    BT    CT    TAT   WT

P1    1     2     18    17    15
P2    2     4     19    17    13
P3    3     6     20    17    11
P4    4     8     21    17    9

Total Turn Around Time is 68
Average Turn Around Time is 17

Total Waiting Time is 48
Average Waiting Time is 12
```


Encrypted_Output.txt - Notepad

```
yrm2j}2k}2l}2}j}2€}33yZ2Z2[2Za2Z`2Z^3y[2[2]2Zb2Z`2Z\3y\2\2_2[Y2Z`2ZZ3y]2]2a2[Z2Z`2b33}˜▯Š•I}ž›–Ij›˜ž–▯I}’–ŽI’œI_a3jŸŽ›Š▯ŽI}ž›–Ij›˜ž–▯
```

- **HRRN :**

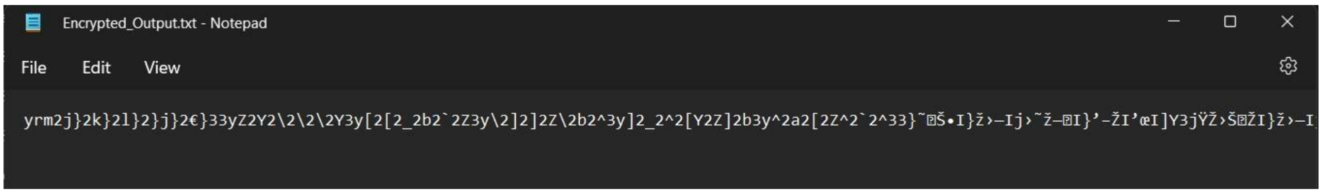


input_9.txt - Notepad

```
5
0 3
2 6
4 4
6 5
8 2
```

Decrypted_output.txt - Notepad

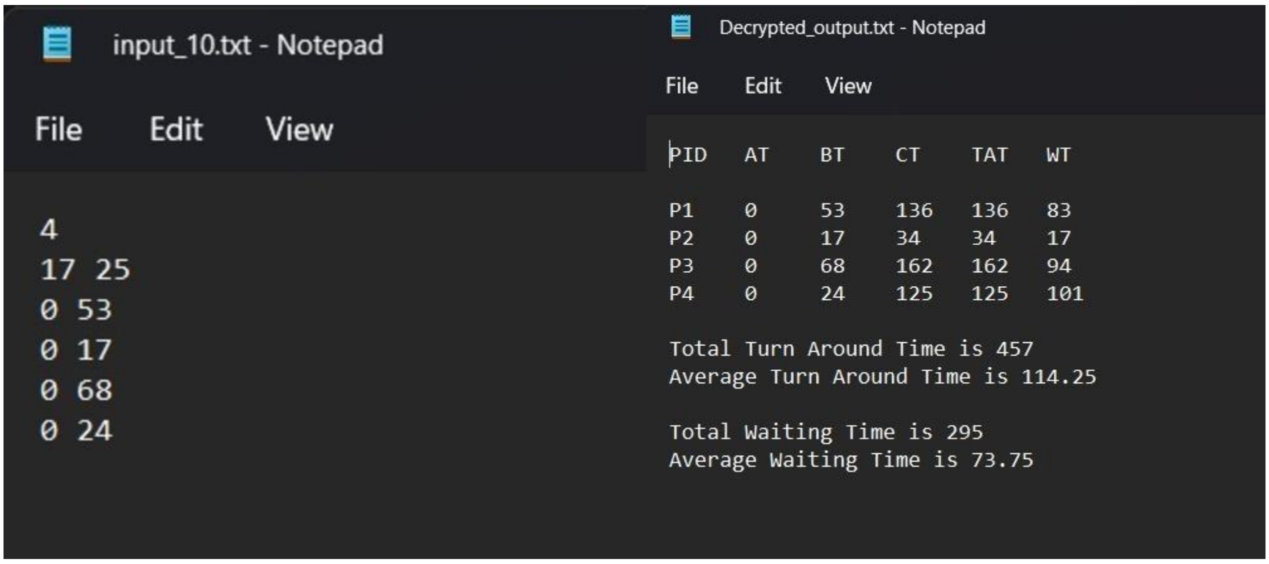
```
PID   AT    BT    CT    TAT   WT

P1    0     3     3     3     0
P2    2     6     9     7     1
P3    4     4     13    9     5
P4    6     5     20    14    9
P5    8     2     15    7     5

Total Turn Around Time is 40
Average Turn Around Time is 8

Total Waiting Time is 20
Average Waiting Time is 4
```



Encrypted_Output.txt - Notepad

```
yrm2j}2k}2l}2}j}2€}33yZ2Y2\2\2\2Y3y[2[2_2b2`2Z3y\2]2]2Z\2b2^3y]2_2^2[Y2Z]2b3y^2a2[2Z^2`2^33}˜Š•I}ž›–Ij›˜ž–I}’–ŽI’œI]Y3jŸŽ›ŠŽI}ž›–I
```

- **MULTILEVEL FEEDBACK QUEUE :**

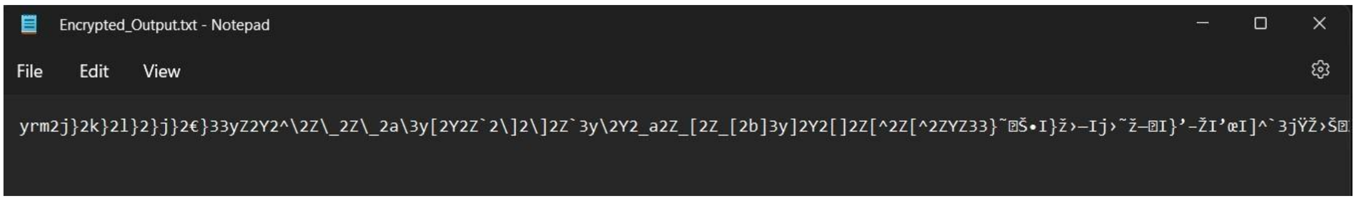


input_10.txt - Notepad

```
4
17 25
0 53
0 17
0 68
0 24
```

Decrypted_output.txt - Notepad

```
PID   AT    BT    CT    TAT   WT

P1    0     53    136   136   83
P2    0     17    34    34    17
P3    0     68    162   162   94
P4    0     24    125   125   101

Total Turn Around Time is 457
Average Turn Around Time is 114.25

Total Waiting Time is 295
Average Waiting Time is 73.75
```

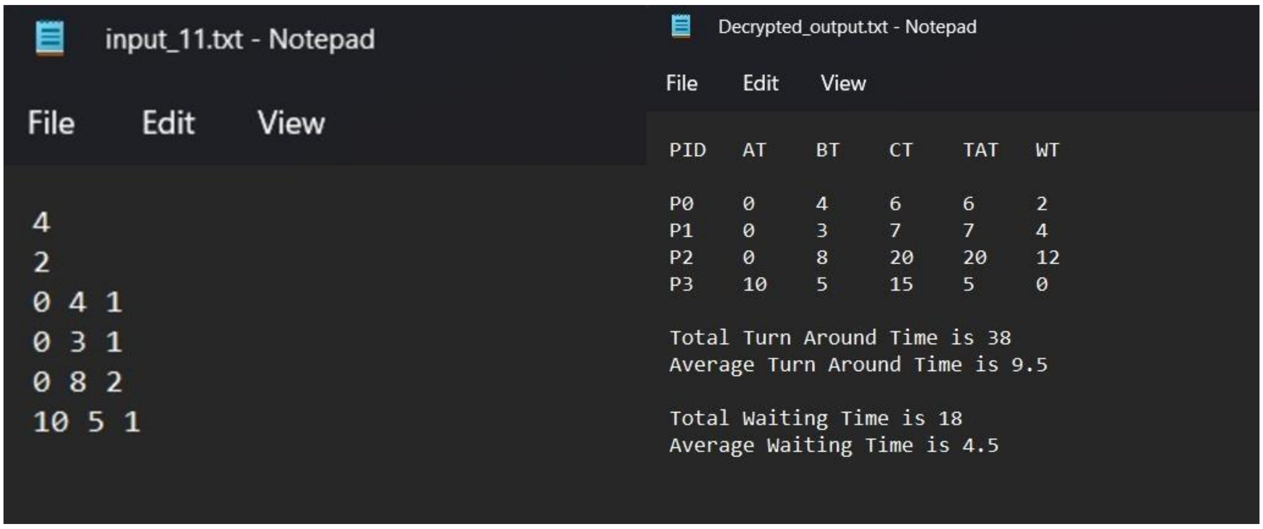


Encrypted_Output.txt - Notepad

```
yrm2j}2k}2l}2}j}2€}33yZ2Y2^\2Z\_2Z\_2a\3y[2Y2Z`2\]2\]2Z`3y\2Y2_a2Z_[2Z_[2b]3y]2Y2[]2Z[^2Z[^2ZYZ33}˜Š•I}ž›–Ij›˜ž–I}’–ŽI’œI]^`3jŸŽ›Š
```

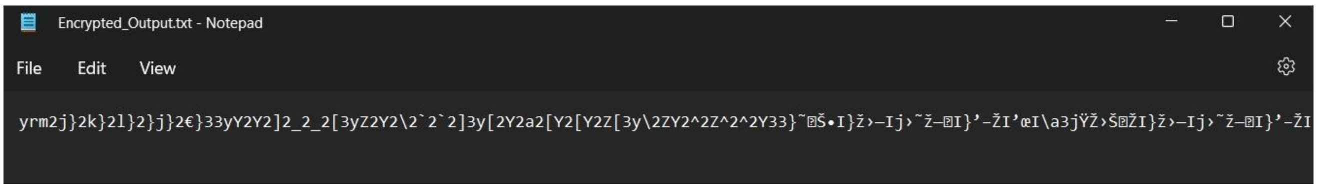
\

- **<u>MULTILEVEL QUEUE</u> :**



```
4
2
0 4 1
0 3 1
0 8 2
10 5 1
```

```
PID   AT   BT   CT   TAT   WT

P0    0    4    6    6     2
P1    0    3    7    7     4
P2    0    8    20   20    12
P3    10   5    15   5     0

Total Turn Around Time is 38
Average Turn Around Time is 9.5

Total Waiting Time is 18
Average Waiting Time is 4.5
```



```
yrm2j}2k}2l}2}j}2€}33yY2Y2]2_2_2[3yZ2Y2\2`2`2]3y[2Y2a2[Y2[Y2Z[3y\2ZY2^2Z^2^2Y33}˜Š•I}ž›–Ij›˜ž–I}’–ŽI’œI\a3jŸŽ›ŠŽI}ž›–Ij›˜ž–I}’–ŽI
```

- ***<u>Conclusion</u>***

In this Project we analysed the process of scheduling and the different scheduling algorithms are reviewed and also presented the basic models of Cryptographic Encryption Techniques . The only reliable technique for evaluating a scheduling algorithm is to implement the algorithm on an actual system and monitor its performance in a *"real-world"* environment.

Drive Link :
<u>OS PROJECT</u>