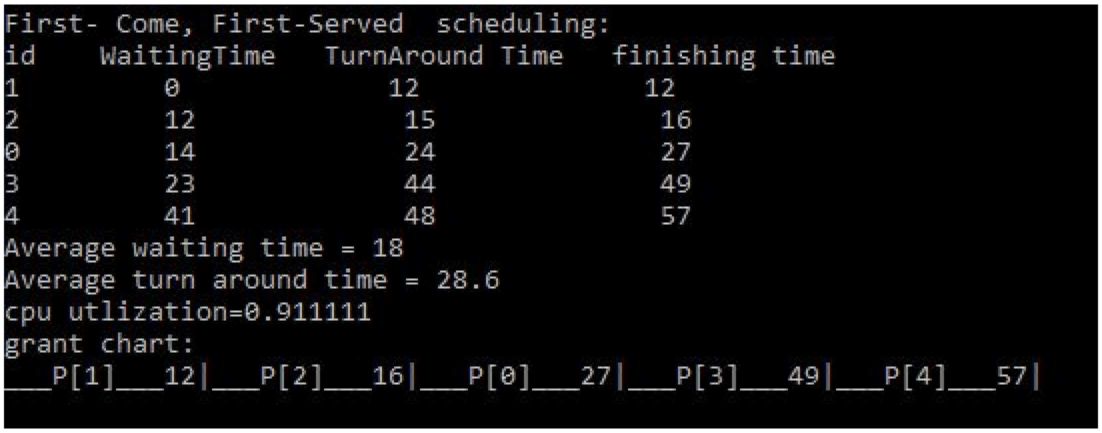
Part 1
We opened a text file named "processes.txt" and then we read the contents of the file and shown it on the screen, the contents of the first four variables Independently and then an array with one dimension called ”a[5]” It was defined by type struct called process which contains {id,arrivetime,brust,size} for each process .
The code for reading the file is placed inside function called read();
The code also contains three functions for each cpu scheduling algorithm (fcfs(), sjf(), rr())
Another 3 one dimension arrays called ”res[5] to put FCFS the result in , s[5] to put SJF the result in, RR[5] to put RR the result in” were defined by a type struct called result which contains {p_num(id of process),waitingt(waiting time for each process),turnaroundt(turn around time for each process)} to put result of each cpu scheduling algorithm in.

In the function of each cpu scheduling algorithm it has been declared how everything has been calculated by writing comments that helps to understand .
How we calculated waiting time ,turn around time,finishing time for each process in the file .
The result for :

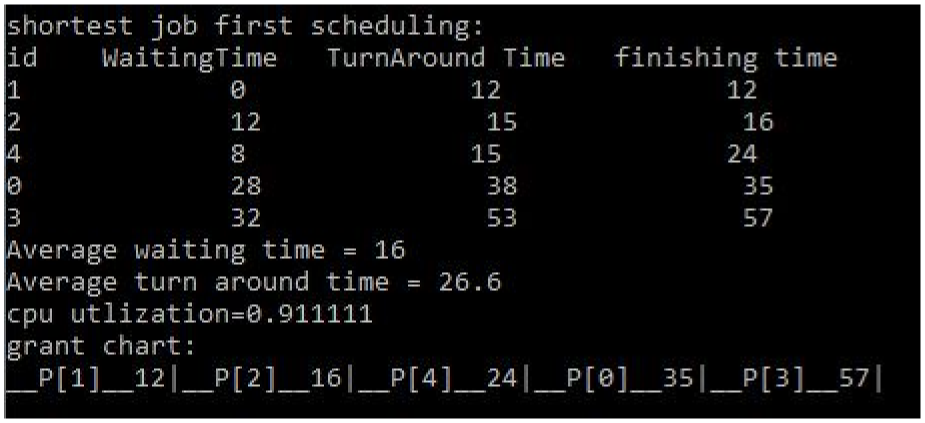
```
4096
512
10
1
0 3 10 8192
1 0 12 2048
2 1 3 512
3 5 21 4096
4 9 7 1024
```

First- Come, First-Served scheduling:



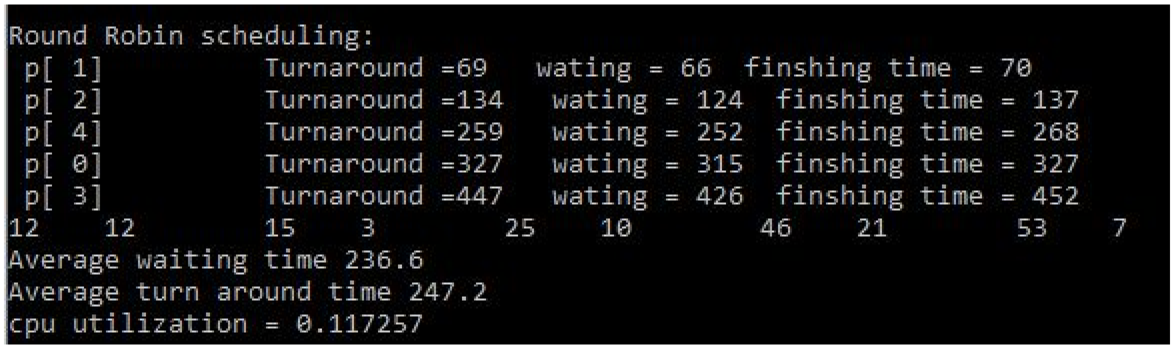
```
First- Come, First-Served  scheduling:
id    WaitingTime   TurnAround Time   finishing time
1        0              12              12
2        12              15              16
0        14              24              27
3        23              44              49
4        41              48              57
Average waiting time = 18
Average turn around time = 28.6
cpu utlization=0.911111
grant chart:
___P[1]___12|___P[2]___16|___P[0]___27|___P[3]___49|___P[4]___57|
```

shortest job first scheduling:



```
shortest job first scheduling:
id    WaitingTime   TurnAround Time   finishing time
1            0              12              12
2            12              15              16
4            8              15              24
0            28              38              35
3            32              53              57
Average waiting time = 16
Average turn around time = 26.6
cpu utlization=0.911111
grant chart:
__P[1]__12|__P[2]__16|__P[4]__24|__P[0]__35|__P[3]__57|
```

Round Robin scheduling:

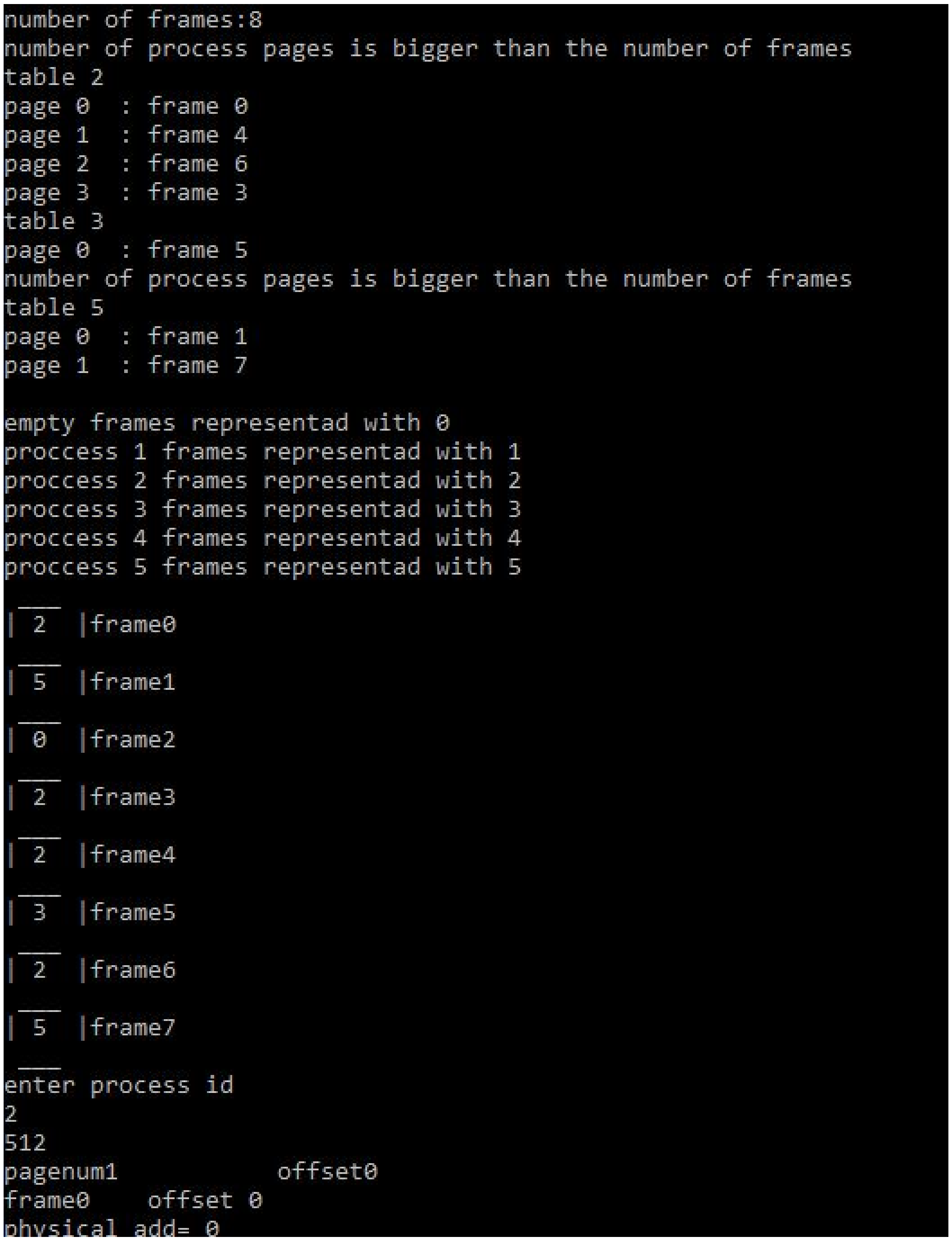


```
Round Robin scheduling:
 p[ 1]          Turnaround =69   wating = 66  finshing time = 70
 p[ 2]          Turnaround =134   wating = 124  finshing time = 137
 p[ 4]          Turnaround =259   wating = 252  finshing time = 268
 p[ 0]          Turnaround =327   wating = 315  finshing time = 327
 p[ 3]          Turnaround =447   wating = 426  finshing time = 452
12    12        15    3        25    10        46    21        53    7
Average waiting time 236.6
Average turn around time 247.2
cpu utilization = 0.117257
```

Part 2:
In the second part of the project we defined physical memory and placed it large enough to accommodate the contents we will add to it so that we don't see errors. We also have 5 processors in the file and each processor has a specific size in the file provided to us. We have created the number of free frames so that they are calculated by dividing the memory size by the page size or frame size. We have kept the initial number of free frames to use later without taking the change that has occurred. We have identified five arrays-like tables for the five processors with us. We found the number of pages per processor and that divided the same size of the processor by the same size as the frame size. We knew the matrix of its name (use) and placed a large block, assuming that it contained the numbers (index frame) so that we gave it an error value that means that these frames have not been used yet. We added (srand(time(0)), because if random numbers are generated we can generate negative numbers and to avoid this problem we have made this addition. After we created a table for each processor, we had to first know the number of pages per processor and compare them to the number of frames in memory. So if the number of pages inside the wizard is greater than the number of frames, the table is not created for this wizard because its page count is greater than the number that I own, If the reverse number of pages for the processor is less than the number of free frames in this case I make the table for the processor. So we did this check and the result was that the number of pages is less than the number of free frames in this case we followed these steps: We worked (for Loop) so that the speaker is less than the number of pages, after which we identified a frame and made it equal to a random number, The first time we came to know the frame and made it equal to a random number and we did (%)the number of frames to make random numbers between 0 and a number entered by the user, we will use the % symbol and which means the remainder) .Then we complete outside of this condition so that this window is not yet used to make the table for the wizard (as shown in the code) after using the window the value for the matrix with a frame number x is used and the value converts to true. For example, we fill out tickets with a X-frame number with a value of "a number to show only where they will appear in memory" and the table is created such that the Y-page table, for example, equals the frame number that will connect me to memory.
We have shown the memory with its "numbers we've added to the memory array within the frequency of table creation," and we show the frame number to its right.” we showed the memory map of the physical memory with filled frames and free frames.”
Our trial system has made us able to accept an interactive logical address from the user and map it to a physical address based on the resulting page allocation in physical frames such as parts a and B above by: We initially identified processor id and made the user the one that he or she entered so that the limited numbers of 0 for 4 are

entered, after which we identified the logical address variable so that it is also entered by the user. We defined the offset variable so that its value remains constant and does not change in both offset in logical address are the same as offset in physical address and we defined a variant of the page number. After entering the processor id in question and entering a logical address by the user, the respective block will go based on the processor id entered. When entering into the block based on the processor id, it checks whether the logical address is less than the number of pages for this wizard by multiplying the page size. If it is less and the condition is met we complete inside the block where the page number is calculated so that the logical address is divided by the page size. Then we move to calculate the offset so that the remainder of the logical address is equal to the page size. After their account we move to calculate the actual address and it is what we want to reach after all these speeches, it is calculated by multiplying a specific frame multiplied by the page size and adding to them offset. Then we showed the shape we wanted to show us in the project.

Result:


```
number of frames:8
number of process pages is bigger than the number of frames
table 2
page 0  : frame 0
page 1  : frame 4
page 2  : frame 6
page 3  : frame 3
table 3
page 0  : frame 5
number of process pages is bigger than the number of frames
table 5
page 0  : frame 1
page 1  : frame 7

empty frames representad with 0
proccess 1 frames representad with 1
proccess 2 frames representad with 2
proccess 3 frames representad with 3
proccess 4 frames representad with 4
proccess 5 frames representad with 5
  ___
| 2  |frame0
  ___
| 5  |frame1
  ___
| 0  |frame2
  ___
| 2  |frame3
  ___
| 2  |frame4
  ___
| 3  |frame5
  ___
| 2  |frame6
  ___
| 5  |frame7
  ___
enter process id
2
512
pagenum1           offset0
frame0    offset 0
physical add= 0
```

Testing file 2 in part 1:

32768

512

  10

  2

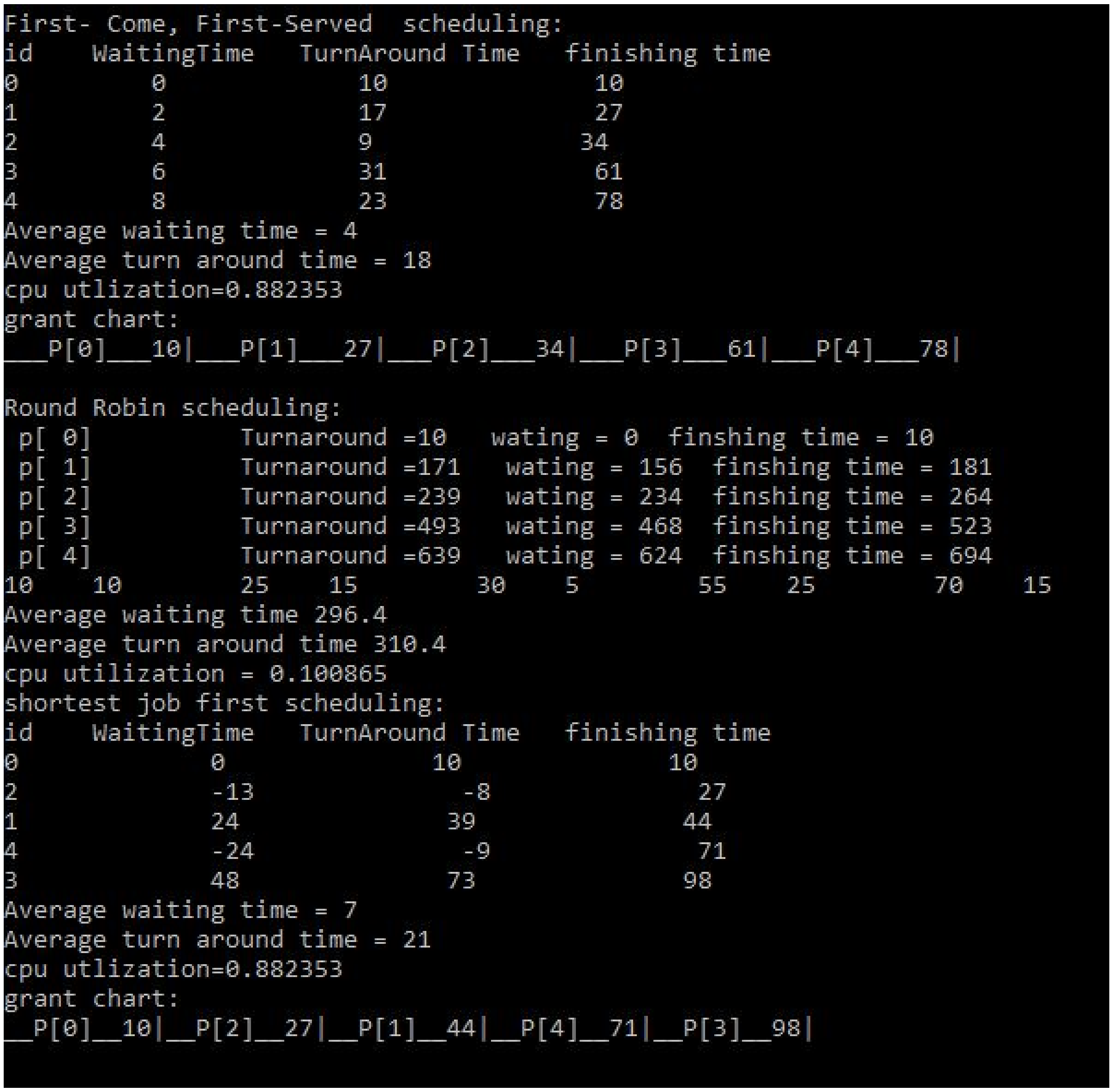
0 0 10 2048

1 10 15 2048

2 25 5 2048

3 30 25 2048

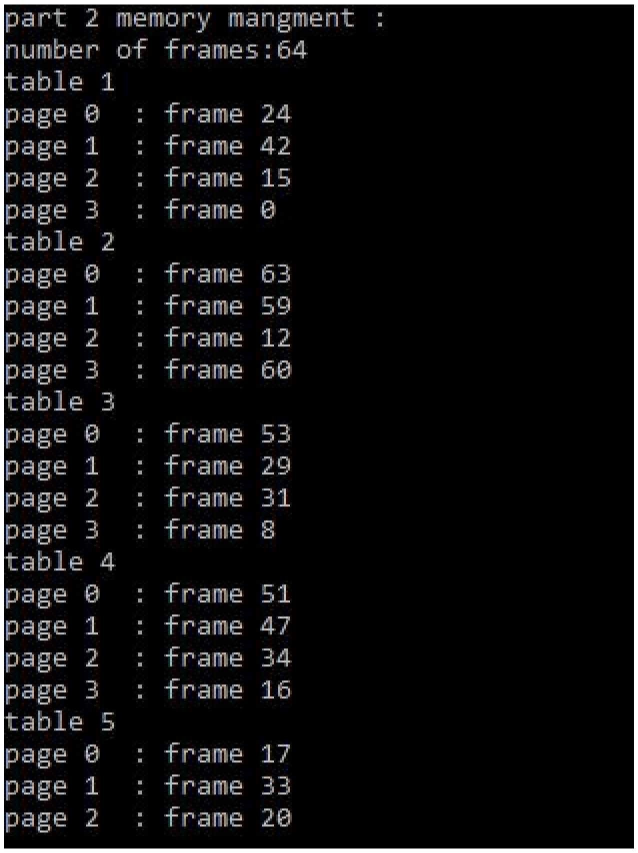
4 55 15 2028



```
First- Come, First-Served  scheduling:
id    WaitingTime   TurnAround Time   finishing time
0        0            10              10
1        2            17              27
2        4            9              34
3        6            31              61
4        8            23              78
Average waiting time = 4
Average turn around time = 18
cpu utlization=0.882353
grant chart:
___P[0]___10|___P[1]___27|___P[2]___34|___P[3]___61|___P[4]___78|

Round Robin scheduling:
 p[ 0]          Turnaround =10   wating = 0  finshing time = 10
 p[ 1]          Turnaround =171   wating = 156  finshing time = 181
 p[ 2]          Turnaround =239   wating = 234  finshing time = 264
 p[ 3]          Turnaround =493   wating = 468  finshing time = 523
 p[ 4]          Turnaround =639   wating = 624  finshing time = 694
10    10        25    15        30    5        55    25        70    15
Average waiting time 296.4
Average turn around time 310.4
cpu utilization = 0.100865
shortest job first scheduling:
id    WaitingTime   TurnAround Time   finishing time
0            0             10              10
2            -13              -8              27
1            24             39              44
4            -24              -9              71
3            48             73              98
Average waiting time = 7
Average turn around time = 21
cpu utlization=0.882353
grant chart:
__P[0]__10|__P[2]__27|__P[1]__44|__P[4]__71|__P[3]__98|
```

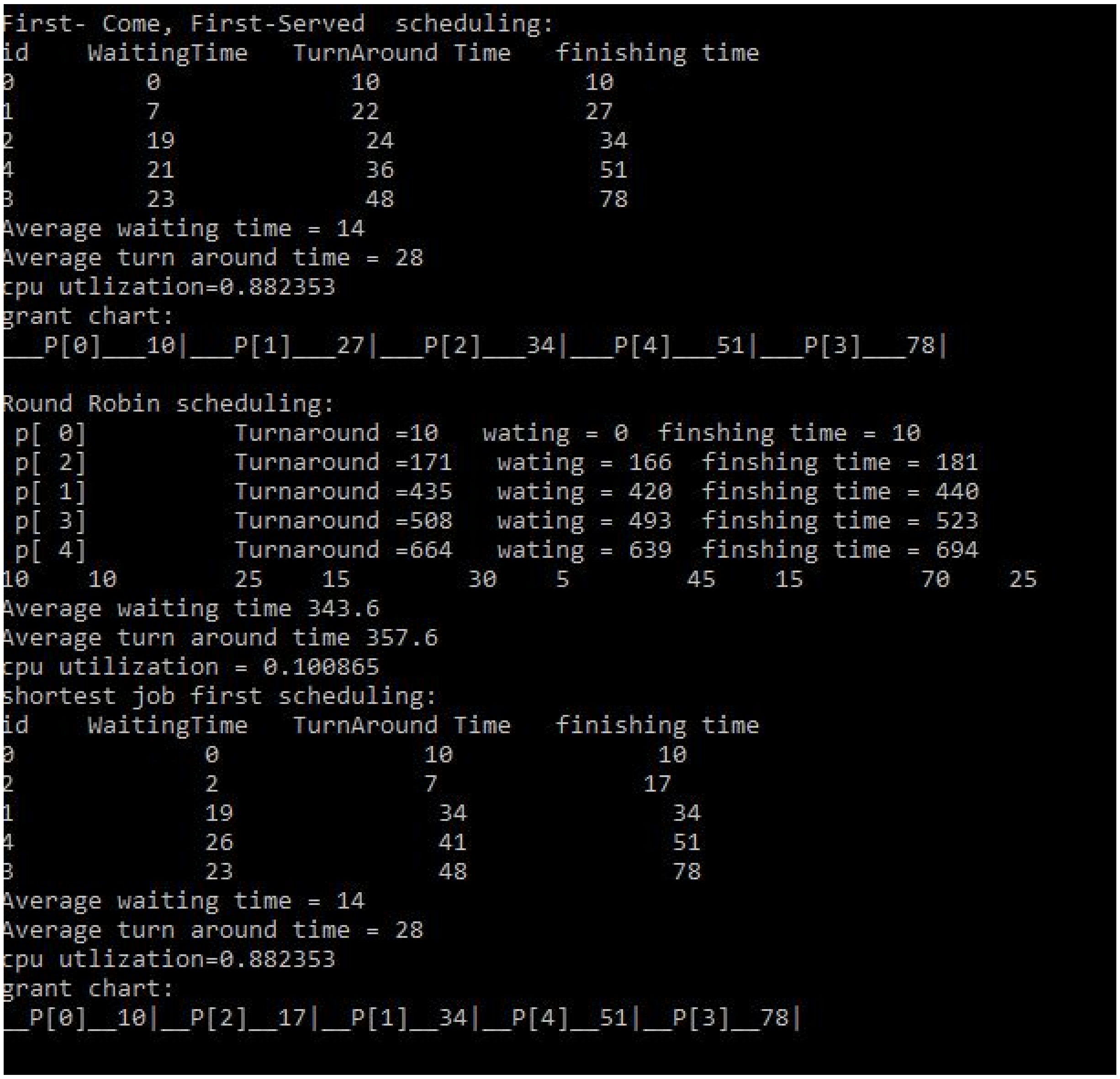
Testing file 2 in part 2:



```
part 2 memory mangment :
number of frames:64
table 1
page 0  : frame 24
page 1  : frame 42
page 2  : frame 15
page 3  : frame 0
table 2
page 0  : frame 63
page 1  : frame 59
page 2  : frame 12
page 3  : frame 60
table 3
page 0  : frame 53
page 1  : frame 29
page 2  : frame 31
page 3  : frame 8
table 4
page 0  : frame 51
page 1  : frame 47
page 2  : frame 34
page 3  : frame 16
table 5
page 0  : frame 17
page 1  : frame 33
page 2  : frame 20
```

Testing file 3 in part 1:

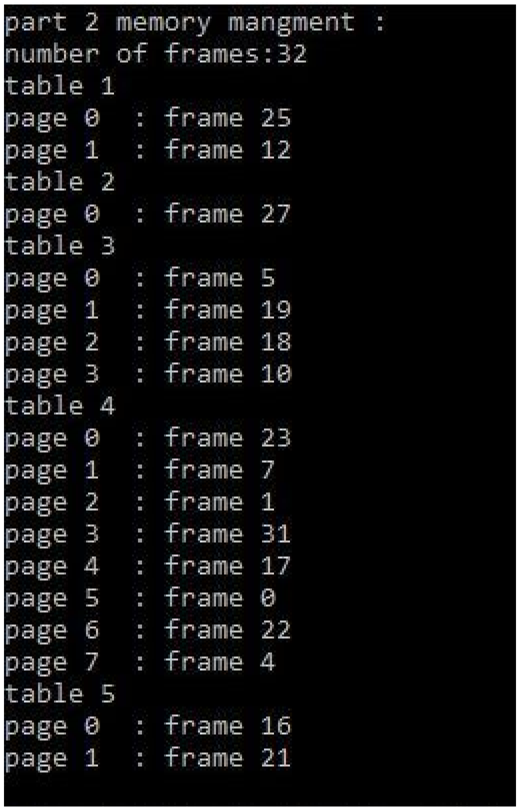
```
65536
2048
  10
  2
0 0 10 4096
1 5 15 2048
2 10  5 8192
3 30 25 16384
4 15 15 4096
```



```
First- Come, First-Served  scheduling:
id    WaitingTime   TurnAround Time   finishing time
0        0             10              10
1        7             22              27
2        19             24              34
4        21             36              51
3        23             48              78
Average waiting time = 14
Average turn around time = 28
cpu utlization=0.882353
grant chart:
___P[0]___10|___P[1]___27|___P[2]___34|___P[4]___51|___P[3]___78|

Round Robin scheduling:
 p[ 0]          Turnaround =10   wating = 0  finshing time = 10
 p[ 2]          Turnaround =171   wating = 166  finshing time = 181
 p[ 1]          Turnaround =435   wating = 420  finshing time = 440
 p[ 3]          Turnaround =508   wating = 493  finshing time = 523
 p[ 4]          Turnaround =664   wating = 639  finshing time = 694
10    10        25    15        30    5        45    15        70    25
Average waiting time 343.6
Average turn around time 357.6
cpu utilization = 0.100865
shortest job first scheduling:
id    WaitingTime   TurnAround Time   finishing time
0            0             10              10
2            2             7              17
1            19             34              34
4            26             41              51
3            23             48              78
Average waiting time = 14
Average turn around time = 28
cpu utlization=0.882353
grant chart:
__P[0]__10|__P[2]__17|__P[1]__34|__P[4]__51|__P[3]__78|
```

Testing file 3 in part 2:



```
part 2 memory mangment :
number of frames:32
table 1
page 0  : frame 25
page 1  : frame 12
table 2
page 0  : frame 27
table 3
page 0  : frame 5
page 1  : frame 19
page 2  : frame 18
page 3  : frame 10
table 4
page 0  : frame 23
page 1  : frame 7
page 2  : frame 1
page 3  : frame 31
page 4  : frame 17
page 5  : frame 0
page 6  : frame 22
page 7  : frame 4
table 5
page 0  : frame 16
page 1  : frame 21
```