

OPERATING SYSTEM LAB ASSIGNMENT – 3

● FCFS

```
• import sys
• sys.stdout = open('output.txt', 'w')
• sys.stdin = open('input.txt' , 'r')
•
• processes = {}
• n = int(input())
• for _ in range(n):
•     s = input().split()
•     del s[-1]
•
•     processes[s[0]] = s[1:]
•
• process_info = {}
• process_state = {}
• input_seq = []
• output_seq = []
• cpu_seq = []
•
• cpu_free = 100
• ip_free = 100
• op_free = 100
• current_time = 0
•
• cpu = []
• inp = []
• out = []
•
• cpu_busy = False
• ip_busy = False
• op_busy = False
•
• while True:
•     for process in list(processes.items()):
•         if int(process[1][0]) <= current_time:
•
•             process_info[process[0]] = process[1]
•             process_state[process[0]] = 2
•
•             cpu.append( (process[0],int(list(process[1][1])[1])) )
•             del processes[process[0]]
```

```

if cpu_free == current_time:
    cpu_busy = False

    temp = process_info[cpu_seq[-3]]
    state = process_state[cpu_seq[-3]]
    if state < len(temp):
        if list(temp[state])[0] == 'I':
            inp.append((cpu_seq[-3],int(list(temp[state])[1])))
        else:
            out.append((cpu_seq[-3],int(list(temp[state])[1])))
        process_state[cpu_seq[-3]] += 1

if not cpu_busy and len(cpu):
    cpu_busy = True
    cpu_seq += [cpu[0][0] , current_time , current_time + cpu[0][1]]
    cpu_free = current_time + cpu[0][1]
    del cpu[0]

if ip_free == current_time:
    ip_busy = False

    temp = process_info[input_seq[-3]]
    state = process_state[input_seq[-3]]
    if state < len(temp):
        if list(temp[state])[0] == 'C':
            cpu.append((input_seq[-3],int(list(temp[state])[1])))
        else:
            out.append((input_seq[-3],int(list(temp[state])[1])))
        process_state[input_seq[-3]] += 1

if not ip_busy and len(inp):
    ip_busy = True
    input_seq += [inp[0][0] , current_time , current_time + inp[0][1]]
    ip_free = current_time + inp[0][1]
    del inp[0]

if op_free == current_time:
    op_busy = False

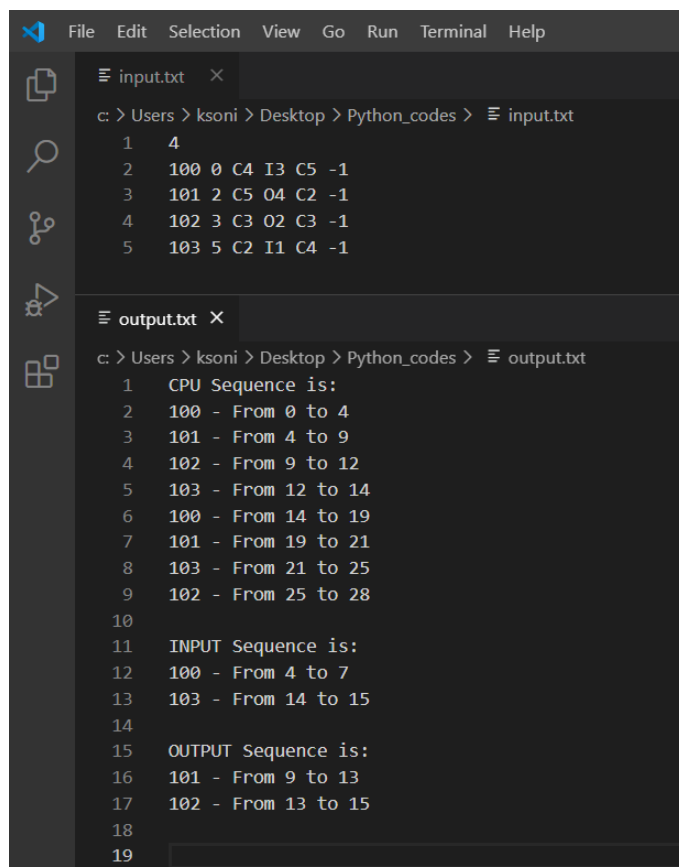
    temp = process_info[output_seq[-3]]
    state = process_state[output_seq[-3]]
    if state < len(temp):
        if list(temp[state])[0] == 'C':
            cpu.append((output_seq[-3],int(list(temp[state])[1])))
        else:
            inp.append((output_seq[-3],int(list(temp[state])[1])))
        process_state[output_seq[-3]] += 1

```

```

•     if not op_busy and len(out):
•         op_busy = True
•         output_seq += [out[0][0] , current_time , current_time + out[0][1]]
•         op_free = current_time + out[0][1]
•         del out[0]
•
•
•     current_time += 1
•     if current_time == 50:
•         break
•
•
• print("CPU Sequence is:")
• for i in range(0,len(cpu_seq),3):
•     print("{} - From {} to {}".format(cpu_seq[i],cpu_seq[i+1],cpu_seq[i+2]))
• print("")
•
• print("INPUT Sequence is:")
• for i in range(0,len(input_seq),3):
•     print("{} - From {} to {}".format(input_seq[i],input_seq[i+1],input_seq[i+2]))
• print("")
•
• print("OUTPUT Sequence is:")
• for i in range(0,len(output_seq),3):
•     print("{} - From {} to {}".format(output_seq[i],output_seq[i+1],output_seq[i+2]))
• print("")
•

```



```

File Edit Selection View Go Run Terminal Help
input.txt X
c: > Users > ksoni > Desktop > Python_codes > input.txt
1 4
2 100 0 C4 I3 C5 -1
3 101 2 C5 04 C2 -1
4 102 3 C3 02 C3 -1
5 103 5 C2 I1 C4 -1

output.txt X
c: > Users > ksoni > Desktop > Python_codes > output.txt
1 CPU Sequence is:
2 100 - From 0 to 4
3 101 - From 4 to 9
4 102 - From 9 to 12
5 103 - From 12 to 14
6 100 - From 14 to 19
7 101 - From 19 to 21
8 103 - From 21 to 25
9 102 - From 25 to 28
10
11 INPUT Sequence is:
12 100 - From 4 to 7
13 103 - From 14 to 15
14
15 OUTPUT Sequence is:
16 101 - From 9 to 13
17 102 - From 13 to 15
18
19

```

• SJF

```
• import sys
• sys.stdout = open('output.txt', 'w')
• sys.stdin = open('input.txt' , 'r')
•
• processes = {}
• n = int(input())
• for _ in range(n):
•     s = input().split()
•     del s[-1]
•
•     processes[s[0]] = s[1:]
•
• process_info = {}
• process_state = {}
• input_seq = []
• output_seq = []
• cpu_seq = []
•
• cpu_free = 100
• ip_free = 100
• op_free = 100
• current_time = 0
•
• cpu = []
• inp = []
• out = []
•
• cpu_busy = False
• ip_busy = False
• op_busy = False
•
• while True:
•     for process in list(processes.items()):
•         if int(process[1][0]) <= current_time:
•
•             process_info[process[0]] = process[1]
•             process_state[process[0]] = 2
•
•             cpu.append( (process[0],int(list(process[1][1])[1])) )
•             del processes[process[0]]
•
•
•     if cpu_free == current_time:
•         cpu_busy = False
•
•         temp = process_info[cpu_seq[-3]]
•         state = process_state[cpu_seq[-3]]
•         if state < len(temp):
•             if list(temp[state])[0] == 'I':
```

```

    inp.append((cpu_seq[-3],int(list(temp[state]))[1])))
    else:
        out.append((cpu_seq[-3],int(list(temp[state]))[1])))
        process_state[cpu_seq[-3]] += 1

if not cpu_busy and len(cpu):
    cpu_busy = True
    cpu = sorted(cpu , key = lambda cpu: cpu[1])
    cpu_seq += [cpu[0][0] , current_time , current_time + cpu[0][1]]
    cpu_free = current_time + cpu[0][1]
    del cpu[0]

if ip_free == current_time:
    ip_busy = False

    temp = process_info[input_seq[-3]]
    state = process_state[input_seq[-3]]
    if state < len(temp):
        if list(temp[state])[0] == 'C':
            cpu.append((input_seq[-3],int(list(temp[state]))[1])))
        else:
            out.append((input_seq[-3],int(list(temp[state]))[1])))
            process_state[input_seq[-3]] += 1

if not ip_busy and len(inp):
    ip_busy = True
    input_seq += [inp[0][0] , current_time , current_time + inp[0][1]]
    ip_free = current_time + inp[0][1]
    del inp[0]

if op_free == current_time:
    op_busy = False

    temp = process_info[output_seq[-3]]
    state = process_state[output_seq[-3]]
    if state < len(temp):
        if list(temp[state])[0] == 'C':
            cpu.append((output_seq[-3],int(list(temp[state]))[1])))
        else:
            inp.append((output_seq[-3],int(list(temp[state]))[1])))
            process_state[output_seq[-3]] += 1

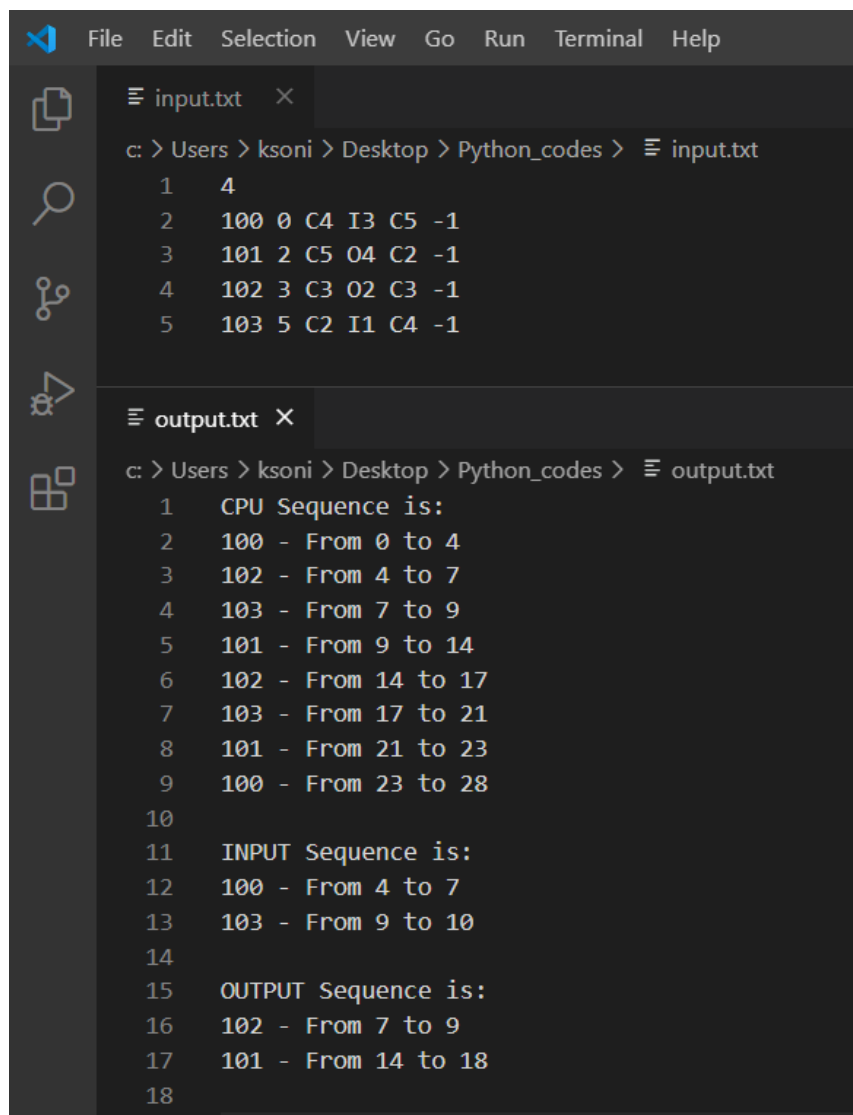
if not op_busy and len(out):
    op_busy = True
    output_seq += [out[0][0] , current_time , current_time + out[0][1]]
    op_free = current_time + out[0][1]
    del out[0]

```

```

•     current_time += 1
•     if current_time == 50:
•         break
•
•
•     print("CPU Sequence is:")
•     for i in range(0,len(cpu_seq),3):
•         print("{} - From {} to {}".format(cpu_seq[i],cpu_seq[i+1],cpu_seq[i+2]))
•     print("")
•
•     print("INPUT Sequence is:")
•     for i in range(0,len(input_seq),3):
•         print("{} - From {} to {}".format(input_seq[i],input_seq[i+1],input_seq[i+2]))
•     print("")
•
•     print("OUTPUT Sequence is:")
•     for i in range(0,len(output_seq),3):
•         print("{} - From {} to {}".format(output_seq[i],output_seq[i+1],output_seq[i+2]))
•     print("")
•

```



The screenshot shows a code editor with two files open: `input.txt` and `output.txt`. The `input.txt` file contains a sequence of 5 lines, each with a number followed by a list of values. The `output.txt` file contains the execution output, showing the CPU sequence, input sequence, and output sequence with their respective ranges.

```

c: > Users > ksoni > Desktop > Python_codes > input.txt
1      4
2    100 0 C4 I3 C5 -1
3    101 2 C5 04 C2 -1
4    102 3 C3 02 C3 -1
5    103 5 C2 I1 C4 -1

c: > Users > ksoni > Desktop > Python_codes > output.txt
1    CPU Sequence is:
2    100 - From 0 to 4
3    102 - From 4 to 7
4    103 - From 7 to 9
5    101 - From 9 to 14
6    102 - From 14 to 17
7    103 - From 17 to 21
8    101 - From 21 to 23
9    100 - From 23 to 28
10
11   INPUT Sequence is:
12   100 - From 4 to 7
13   103 - From 9 to 10
14
15   OUTPUT Sequence is:
16   102 - From 7 to 9
17   101 - From 14 to 18
18

```

• Priority (non – preemtive)

```
• import sys
• sys.stdout = open('output.txt', 'w')
• sys.stdin = open('input.txt' , 'r')
•
• processes = {}
• n = int(input())
• for _ in range(n):
•     s = input().split()
•     del s[-1]
•
•     processes[s[0]] = s[1:]
•
• process_info = {}
• process_state = {}
• priority = {}
• input_seq = []
• output_seq = []
• cpu_seq = []
•
• cpu_free = 100
• ip_free = 100
• op_free = 100
• current_time = 0
•
• cpu = []
• inp = []
• out = []
•
• cpu_busy = False
• ip_busy = False
• op_busy = False
•
• while True:
•     for process in list(processes.items()):
•         if int(process[1][0]) <= current_time:
•
•             process_info[process[0]] = process[1]
•             process_state[process[0]] = 3
•             priority[process[0]] = process[1][1]
•
•             cpu.append( (process[0],int(list(process[1][2]))[1]) , priority[process[0]]) )
•             del processes[process[0]]
•
•         if cpu_free == current_time:
•             cpu_busy = False
•
•             temp = process_info[cpu_seq[-3]]
```

```

state = process_state[cpu_seq[-3]]
if state < len(temp):
    if list(temp[state])[0] == 'I':
        inp.append((cpu_seq[-3],int(list(temp[state]))[1])))
    else:
        out.append((cpu_seq[-3],int(list(temp[state]))[1])))
    process_state[cpu_seq[-3]] += 1

if not cpu_busy and len(cpu):
    cpu_busy = True
    cpu = sorted(cpu , key = lambda cpu: cpu[2])
    cpu_seq += [cpu[0][0] , current_time , current_time + cpu[0][1]]
    cpu_free = current_time + cpu[0][1]
    del cpu[0]

if ip_free == current_time:
    ip_busy = False

    temp = process_info[input_seq[-3]]
    state = process_state[input_seq[-3]]
    if state < len(temp):
        if list(temp[state])[0] == 'C':
            cpu.append((input_seq[-3],int(list(temp[state]))[1]),priority[input_seq[-3]]))
        else:
            out.append((input_seq[-3],int(list(temp[state]))[1])))
            process_state[input_seq[-3]] += 1

if not ip_busy and len(inp):
    ip_busy = True
    input_seq += [inp[0][0] , current_time , current_time + inp[0][1]]
    ip_free = current_time + inp[0][1]
    del inp[0]

if op_free == current_time:
    op_busy = False

    temp = process_info[output_seq[-3]]
    state = process_state[output_seq[-3]]
    if state < len(temp):
        if list(temp[state])[0] == 'C':
            cpu.append((output_seq[-3],int(list(temp[state]))[1]),priority[output_seq[-3]]))
        else:
            inp.append((output_seq[-3],int(list(temp[state]))[1])))
            process_state[output_seq[-3]] += 1

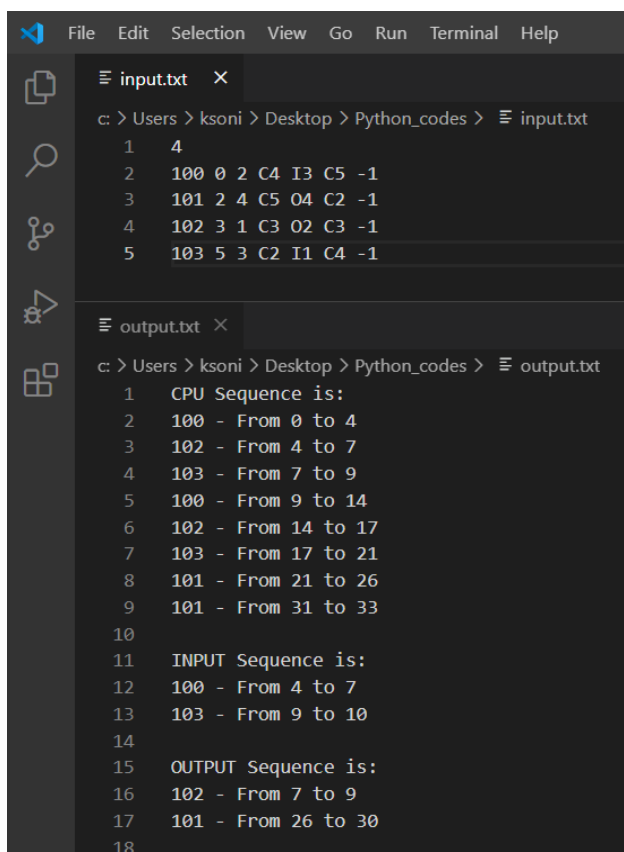
```



```

•     if not op_busy and len(out):
•         op_busy = True
•         output_seq += [out[0][0] , current_time , current_time + out[0][1]]
•         op_free = current_time + out[0][1]
•         del out[0]
•
•     current_time += 1
•     if current_time == 50:
•         break
•
• print("CPU Sequence is:")
• for i in range(0,len(cpu_seq),3):
•     print("{} - From {} to {}".format(cpu_seq[i],cpu_seq[i+1],cpu_seq[i+2]))
• print("")
•
• print("INPUT Sequence is:")
• for i in range(0,len(input_seq),3):
•     print("{} - From {} to {}".format(input_seq[i],input_seq[i+1],input_seq[i+2]))
• print("")
•
• print("OUTPUT Sequence is:")
• for i in range(0,len(output_seq),3):
•     print("{} - From {} to {}".format(output_seq[i],output_seq[i+1],output_seq[i+2]))
• print("")
•

```



The screenshot shows a code editor with two files open: `input.txt` and `output.txt`. The `input.txt` file contains a list of 5 tasks, each with a task ID, a start time, a duration, and a priority. The `output.txt` file shows the CPU sequence and the output sequence for each task, along with the input sequence.

```

c: > Users > ksoni > Desktop > Python_codes > input.txt
1 4
2 100 0 2 C4 I3 C5 -1
3 101 2 4 C5 O4 C2 -1
4 102 3 1 C3 O2 C3 -1
5 103 5 3 C2 I1 C4 -1

c: > Users > ksoni > Desktop > Python_codes > output.txt
1 CPU Sequence is:
2 100 - From 0 to 4
3 102 - From 4 to 7
4 103 - From 7 to 9
5 100 - From 9 to 14
6 102 - From 14 to 17
7 103 - From 17 to 21
8 101 - From 21 to 26
9 101 - From 31 to 33
10
11 INPUT Sequence is:
12 100 - From 4 to 7
13 103 - From 9 to 10
14
15 OUTPUT Sequence is:
16 102 - From 7 to 9
17 101 - From 26 to 30
18

```

• Preemptive Priority

```
• import sys
• sys.stdout = open('output.txt', 'w')
• sys.stdin = open('input.txt' , 'r')
•
• processes = {}
• n = int(input())
• for _ in range(n):
•     s = input().split()
•     del s[-1]
•
•     processes[s[0]] = s[1:]
•
• process_info = {}
• process_state = {}
• priority = {}
• input_seq = []
• output_seq = []
• cpu_seq = []
•
• cpu_free = 100
• ip_free = 100
• op_free = 100
• current_time = 0
•
• cpu = []
• inp = []
• out = []
•
• cpu_busy = False
• ip_busy = False
• op_busy = False
•
• while len(processes) or len(cpu) or len(inp) or len(out) or True:
•     flag = 0
•     for process in list(processes.items()):
•         if int(process[1][0]) <= current_time:
•
•             process_info[process[0]] = process[1]
•             process_state[process[0]] = 3
•             priority[process[0]] = process[1][1]
•
•             flag = 1
•             cpu.append( (process[0],int(list(process[1][2]))[1]) , priority[process[0]]) )
•             del processes[process[0]]
•
•             if cpu_free == current_time:
•                 cpu_busy = False
```

```


•   cpu_seq.append(current_time)
•   cpu = sorted(cpu , key = lambda cpu: cpu[2])
•   del cpu[0]
•
•
•   temp = process_info[cpu_seq[-3]]
•   state = process_state[cpu_seq[-3]]
•   if state < len(temp):
•       if list(temp[state])[0] == 'I':
•           inp.append((cpu_seq[-3],int(list(temp[state]))[1]))
•       else:
•           out.append((cpu_seq[-3],int(list(temp[state]))[1]))
•           process_state[cpu_seq[-3]] += 1
•
•
•
•
•   if ip_free == current_time:
•       ip_busy = False
•
•
•       temp = process_info[input_seq[-3]]
•       state = process_state[input_seq[-3]]
•       if state < len(temp):
•           if list(temp[state])[0] == 'C':
•               flag = 1
•               cpu.append((input_seq[-3],int(list(temp[state]))[1]),prior-
ity[input_seq[-3]])
•           else:
•               out.append((input_seq[-3],int(list(temp[state]))[1]))
•               process_state[input_seq[-3]] += 1
•
•
•   if not ip_busy and len(inp):
•       ip_busy = True
•       input_seq += [inp[0][0] , current_time , current_time + inp[0][1]]
•       ip_free = current_time + inp[0][1]
•       del inp[0]
•
•
•
•
•   if op_free == current_time:
•       op_busy = False
•
•
•       temp = process_info[output_seq[-3]]
•       state = process_state[output_seq[-3]]
•       if state < len(temp):
•           if list(temp[state])[0] == 'C':
•               flag = 1
•               cpu.append((output_seq[-3],int(list(temp[state]))[1]),prior-
ity[output_seq[-3]])
•           else:
•               inp.append((output_seq[-3],int(list(temp[state]))[1]))
•               process_state[output_seq[-3]] += 1
•
•
•   if not op_busy and len(out):






```

```

•         op_busy = True
•         output_seq += [out[0][0] , current_time , current_time + out[0][1]]
•         op_free = current_time + out[0][1]
•         del out[0]
•
•
•
•
•         if flag == 1 and cpu_busy:
•             sth = sorted(cpu , key = lambda cpu: cpu[2])
•             if cpu_seq[-2] != sth[0][2]:
•                 cpu_seq.append(current_time)
•                 del cpu[0]
•                 cpu.append((cpu_seq[-3] , cpu_free - current_time , priority[cpu_seq[-3]]))
•                 cpu_free = current_time + sth[0][1]
•                 cpu_seq += [sth[0][0] , current_time]
•
•             elif not cpu_busy and len(cpu):
•                 cpu_busy = True
•                 cpu = sorted(cpu , key = lambda cpu: cpu[2])
•                 cpu_seq += [cpu[0][0] , current_time]
•                 cpu_free = current_time + cpu[0][1]
•
•         current_time += 1
•         if current_time == 50:
•             break
•
•     print("CPU Sequence is:")
•     for i in range(0,len(cpu_seq),3):
•         print("{} - From {} to {}".format(cpu_seq[i],cpu_seq[i+1],cpu_seq[i+2]))
•     print("")
•
•     print("INPUT Sequence is:")
•     for i in range(0,len(input_seq),3):
•         print("{} - From {} to {}".format(input_seq[i],input_seq[i+1],input_seq[i+2]))
•     print("")
•
•     print("OUTPUT Sequence is:")
•     for i in range(0,len(output_seq),3):
•         print("{} - From {} to {}".format(output_seq[i],output_seq[i+1],output_seq[i+2]))
•     print("")
•

```

FileEditSelectionViewGoRunTerminalHelp



≡ input.txt ×

c: > Users > ksoni > Desktop > Python_codes > ≡ input.txt

14

2100 0 2 C4 I3 C5 -1

3101 2 4 C5 04 C2 -1

4102 3 1 C3 02 C3 -1

5103 5 3 C2 I1 C4 -1

≡ output.txt ×

c: > Users > ksoni > Desktop > Python_codes > ≡ output.txt

1CPU Sequence is:

2100 - From 0 to 2

3100 - From 2 to 3

4102 - From 3 to 5

5102 - From 5 to 8

6102 - From 8 to 9

7100 - From 9 to 12

8103 - From 12 to 14

9100 - From 15 to 16

10100 - From 16 to 21

11103 - From 21 to 25

12

13INPUT Sequence is:

14100 - From 12 to 15

15103 - From 15 to 16

16

17OUTPUT Sequence is:

18102 - From 8 to 10

19102 - From 10 to 13

20

21

• MLFQ

```
• import sys
• sys.stdout = open('output.txt', 'w')
• sys.stdin = open('input.txt' , 'r')
•
• processes = {}
• n = int(input())
• for _ in range(n):
•     s = input().split()
•     del s[-1]
•
•     processes[s[0]] = s[1:]
•
• process_info = {}
• process_state = {}
•
• input_seq = []
• output_seq = []
• cpu_seq = []
•
• cpu_free = 100
• ip_free = 100
• op_free = 100
• current_time = 0
•
• work = 1000000
• cpu1 = []
• cpu2 = []
• cpu3 = []
• inp = []
• out = []
•
• cpu_busy = False
• ip_busy = False
• op_busy = False
•
• while True:
•     for process in list(processes.items()):
•         if int(process[1][0]) <= current_time:
•
•             process_info[process[0]] = process[1]
•             process_state[process[0]] = 2
•
•             cpu1.append( (process[0],int(list(process[1][1]))[1]) )
•             del processes[process[0]]
•
•         if cpu_free == current_time:
•             cpu_busy = False
•             cpu_seq.append(current_time)
```

```

•
•
temp = process_info[cpu_seq[-3]]
state = process_state[cpu_seq[-3]]
if state < len(temp):
•
    if list(temp[state])[0] == 'I':
•
        inp.append((cpu_seq[-3],int(list(temp[state])[1])))
•
    else:
•
        out.append((cpu_seq[-3],int(list(temp[state])[1])))
•
        process_state[cpu_seq[-3]] += 1
•

•

if ip_free == current_time:
•
    ip_busy = False
•
•
temp = process_info[input_seq[-3]]
state = process_state[input_seq[-3]]
if state < len(temp):
•
    if list(temp[state])[0] == 'C':
•
        cpu1.append((input_seq[-3],int(list(temp[state])[1])))
•
    else:
•
        out.append((input_seq[-3],int(list(temp[state])[1])))
•
        process_state[input_seq[-3]] += 1
•

•

if not ip_busy and len(inp):
•
    ip_busy = True
•
    input_seq += [inp[0][0] , current_time , current_time + inp[0][1]]
•
    ip_free = current_time + inp[0][1]
•
    del inp[0]
•

•

if op_free == current_time:
•
    op_busy = False
•
•
temp = process_info[output_seq[-3]]
state = process_state[output_seq[-3]]
if state < len(temp):
•
    if list(temp[state])[0] == 'C':
•
        cpu1.append((output_seq[-3],int(list(temp[state])[1])))
•
    else:
•
        inp.append((output_seq[-3],int(list(temp[state])[1])))
•
        process_state[output_seq[-3]] += 1
•

•

if not op_busy and len(out):
•
    op_busy = True
•
    output_seq += [out[0][0] , current_time , current_time + out[0][1]]
•
    op_free = current_time + out[0][1]
•
    del out[0]
•

```

```

• if not cpu_busy and len(cpu1):
•     cpu_busy = True
•     cpu_seq += [cpu1[0][0] , current_time]
•     cpu_free = current_time + cpu1[0][1]
•     work = 1
•     del cpu1[0]
•
• elif len(cpu1) and cpu_busy:
•     cpu_seq.append(current_time)
•     cpu2.append((cpu_seq[-3] , cpu_free - current_time ))
•
•     cpu_seq += [cpu1[0][0] , current_time]
•     cpu_free = current_time + cpu1[0][1]
•     del cpu1[0]
•
• elif len(cpu2) and not cpu_busy:
•     cpu_busy = True
•     work = 2
•     cpu_seq += [cpu2[0][0] , current_time]
•     cpu_free = current_time + cpu2[0][1]
•     del cpu2[0]
•
• elif len(cpu2) and cpu_busy and work != 1:
•     cpu_seq.append(current_time)
•     cpu3.append((cpu_seq[-3] , cpu_free - current_time ))
•
•     cpu_seq += [cpu2[0][0] , current_time]
•     cpu_free = current_time + cpu2[0][1]
•     del cpu2[0]
•
• elif len(cpu3) and not cpu_busy:
•     cpu_busy = True
•     work = 3
•     cpu_seq += [cpu3[0][0] , current_time]
•     cpu_free = current_time + cpu3[0][1]
•     del cpu3[0]
•
• if current_time % 10 == 0:
•     cpu1 += cpu2 + cpu3
•     cpu2 = []
•     cpu3 = []
•
• current_time += 1
• if current_time == 50:
•     break
•
• print("CPU Sequence is:")

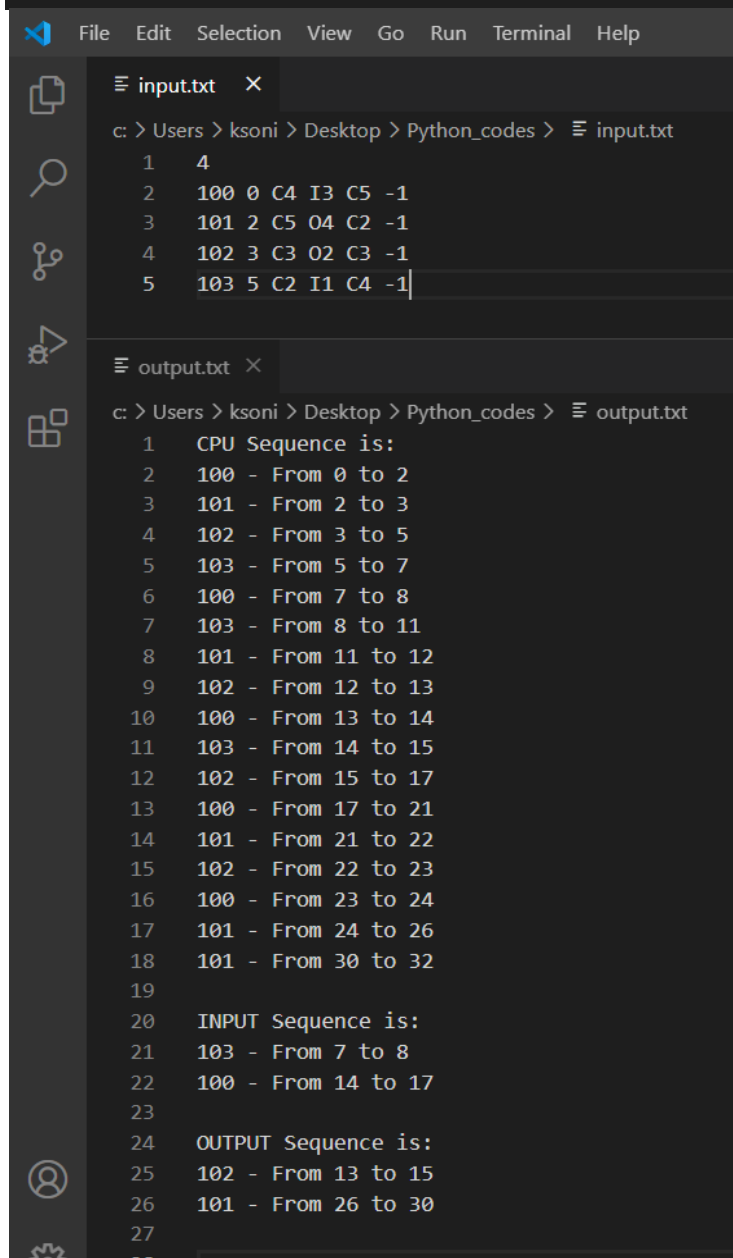
```



```

• for i in range(0,len(cpu_seq),3):
•     print("{} - From {} to {}".format(cpu_seq[i],cpu_seq[i+1],cpu_seq[i+2]))
• print("")
•
• print("INPUT Sequence is:")
• for i in range(0,len(input_seq),3):
•     print("{} - From {} to {}".format(input_seq[i],input_seq[i+1],input_seq[i+2]))
• print("")
•
• print("OUTPUT Sequence is:")
• for i in range(0,len(output_seq),3):
•     print("{} - From {} to {}".format(output_seq[i],output_seq[i+1],output_seq[i+2]))
• print("")
•

```



The screenshot shows a VS Code editor with two files open: `input.txt` and `output.txt`. The `input.txt` file contains the following content:

```

1 4
2 100 0 C4 I3 C5 -1
3 101 2 C5 04 C2 -1
4 102 3 C3 02 C3 -1
5 103 5 C2 I1 C4 -1

```

The `output.txt` file shows the execution of a Python script, displaying the following content:

```

1 CPU Sequence is:
2 100 - From 0 to 2
3 101 - From 2 to 3
4 102 - From 3 to 5
5 103 - From 5 to 7
6 100 - From 7 to 8
7 103 - From 8 to 11
8 101 - From 11 to 12
9 102 - From 12 to 13
10 100 - From 13 to 14
11 103 - From 14 to 15
12 102 - From 15 to 17
13 100 - From 17 to 21
14 101 - From 21 to 22
15 102 - From 22 to 23
16 100 - From 23 to 24
17 101 - From 24 to 26
18 101 - From 30 to 32
19
20 INPUT Sequence is:
21 103 - From 7 to 8
22 100 - From 14 to 17
23
24 OUTPUT Sequence is:
25 102 - From 13 to 15
26 101 - From 26 to 30
27
28

```

• Round Robin

```
• import sys
• sys.stdout = open('output.txt', 'w')
• sys.stdin = open('input.txt' , 'r')
•
• processes = {}
• n = int(input())
• for _ in range(n):
•     s = input().split()
•     del s[-1]
•
•     processes[s[0]] = s[1:]
•
• process_info = {}
• process_state = {}
•
• input_seq = []
• output_seq = []
• cpu_seq = []
•
• cpu_free = 100
• ip_free = 100
• op_free = 100
• current_time = 0
•
• cpu = []
• inp = []
• out = []
•
• cpu_busy = False
• ip_busy = False
• op_busy = False
•
• while True:
•     for process in list(processes.items()):
•         if int(process[1][0]) <= current_time:
•
•             process_info[process[0]] = process[1]
•             process_state[process[0]] = 2
•
•             cpu.append( (process[0],int(list(process[1][1]))[1]) )
•             del processes[process[0]]
•
•
•     if cpu_free == current_time:
•         cpu_busy = False
•         cpu_seq.append(current_time)
•
•
•         temp = process_info[cpu_seq[-3]]
•         state = process_state[cpu_seq[-3]]

```

```

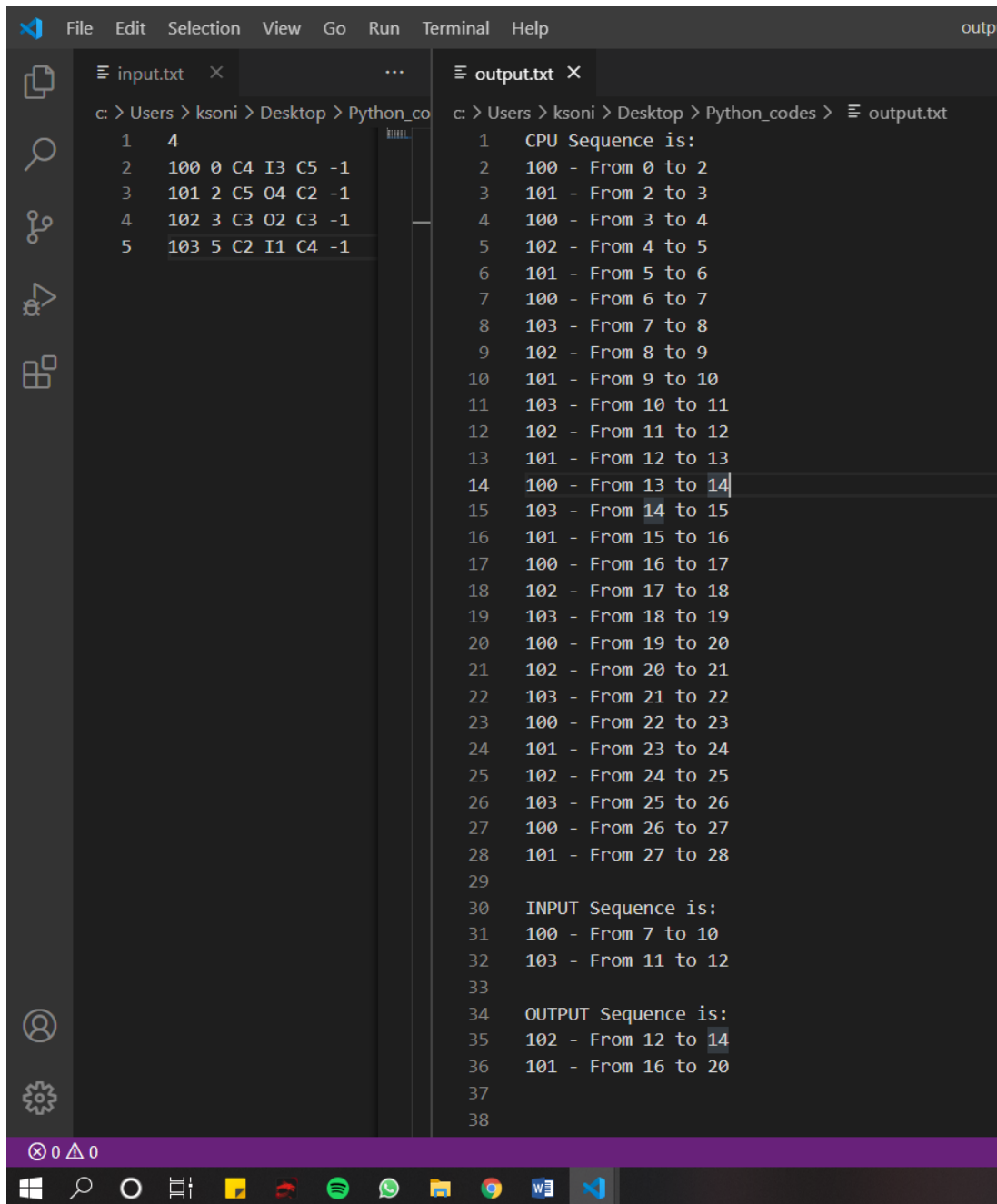
•         if state < len(temp):
•             if list(temp[state])[0] == 'I':
•                 inp.append((cpu_seq[-3],int(list(temp[state])[1])))
•             else:
•                 out.append((cpu_seq[-3],int(list(temp[state])[1])))
•                 process_state[cpu_seq[-3]] += 1
•
•
•
•
•         if len(cpu) and cpu_busy:
•             cpu_seq.append(current_time)
•             cpu.append((cpu_seq[-3] , cpu_free - current_time ))
•
•
•             cpu_seq += [cpu[0][0] , current_time]
•             cpu_free = current_time + cpu[0][1]
•             del cpu[0]
•
•
•         if not cpu_busy and len(cpu):
•             cpu_busy = True
•             cpu_seq += [cpu[0][0] , current_time]
•             cpu_free = current_time + cpu[0][1]
•             del cpu[0]
•
•
•         if ip_free == current_time:
•             ip_busy = False
•
•
•             temp = process_info[input_seq[-3]]
•             state = process_state[input_seq[-3]]
•             if state < len(temp):
•                 if list(temp[state])[0] == 'C':
•                     cpu.append((input_seq[-3],int(list(temp[state])[1])))
•                 else:
•                     out.append((input_seq[-3],int(list(temp[state])[1])))
•                     process_state[input_seq[-3]] += 1
•
•
•         if not ip_busy and len(inp):
•             ip_busy = True
•             input_seq += [inp[0][0] , current_time , current_time + inp[0][1]]
•             ip_free = current_time + inp[0][1]
•             del inp[0]
•
•
•
•
•
•
•
•
•         if op_free == current_time:
•             op_busy = False
•
•
•             temp = process_info[output_seq[-3]]
•             state = process_state[output_seq[-3]]
•             if state < len(temp):
•                 if list(temp[state])[0] == 'C':
•                     cpu.append((output_seq[-3],int(list(temp[state])[1])))
•                 else:
•                     inp.append((output_seq[-3],int(list(temp[state])[1])))

```

```

•     process_state[output_seq[-3]] += 1
•
•     if not op_busy and len(out):
•         op_busy = True
•         output_seq += [out[0][0] , current_time , current_time + out[0][1]]
•         op_free = current_time + out[0][1]
•         del out[0]
•
•     current_time += 1
•     if current_time == 50:
•         break
•
• print("CPU Sequence is:")
• for i in range(0,len(cpu_seq),3):
•     print("{} - From {} to {}".format(cpu_seq[i],cpu_seq[i+1],cpu_seq[i+2]))
• print("")
•
• print("INPUT Sequence is:")
• for i in range(0,len(input_seq),3):
•     print("{} - From {} to {}".format(input_seq[i],input_seq[i+1],input_seq[i+2]))
• print("")
•
• print("OUTPUT Sequence is:")
• for i in range(0,len(output_seq),3):
•     print("{} - From {} to {}".format(output_seq[i],output_seq[i+1],output_seq[i+2]))
• print("")
•

```



```
File Edit Selection View Go Run Terminal Help
input.txt x output.txt x
c: > Users > ksoni > Desktop > Python_codes > output.txt
1 4
2 100 0 C4 I3 C5 -1
3 101 2 C5 04 C2 -1
4 102 3 C3 02 C3 -1
5 103 5 C2 I1 C4 -1
1 CPU Sequence is:
2 100 - From 0 to 2
3 101 - From 2 to 3
4 100 - From 3 to 4
5 102 - From 4 to 5
6 101 - From 5 to 6
7 100 - From 6 to 7
8 103 - From 7 to 8
9 102 - From 8 to 9
10 101 - From 9 to 10
11 103 - From 10 to 11
12 102 - From 11 to 12
13 101 - From 12 to 13
14 100 - From 13 to 14
15 103 - From 14 to 15
16 101 - From 15 to 16
17 100 - From 16 to 17
18 102 - From 17 to 18
19 103 - From 18 to 19
20 100 - From 19 to 20
21 102 - From 20 to 21
22 103 - From 21 to 22
23 100 - From 22 to 23
24 101 - From 23 to 24
25 102 - From 24 to 25
26 103 - From 25 to 26
27 100 - From 26 to 27
28 101 - From 27 to 28
29
30 INPUT Sequence is:
31 100 - From 7 to 10
32 103 - From 11 to 12
33
34 OUTPUT Sequence is:
35 102 - From 12 to 14
36 101 - From 16 to 20
37
38
```

• Lottery (proportional share)

```
• import sys
• sys.stdout = open('output.txt', 'w')
• sys.stdin = open('input.txt', 'r')
• import random
•
• processes = {}
• n = int(input())
• for _ in range(n):
```

```

•     s = input().split()
•     del s[-1]
•
•     processes[s[0]] = s[1:]
•
•     process_info = {}
•     process_state = {}
•
•     input_seq = []
•     output_seq = []
•     cpu_seq = []
•     rand = []
•
•     cpu_free = 100
•     ip_free = 100
•     op_free = 100
•     current_time = 0
•
•     cpu = {}
•     inp = []
•     out = []
•
•     cpu_busy = False
•     ip_busy = False
•     op_busy = False
•
•     while True:
•         for process in list(processes.items()):
•             if int(process[1][0]) <= current_time:
•                 rand += [process[0] for i in range(int(process[1][1]))]
•                 process_info[process[0]] = process[1]
•                 process_state[process[0]] = 3
•
•                 cpu[process[0]] = int(list(process[1][2])[1])
•                 del processes[process[0]]
•
•         random.shuffle(rand)
•
•         if cpu_free == current_time:
•             cpu_busy = False
•             cpu_seq.append(current_time)
•             del cpu[cpu_seq[-3]]
•             while cpu_seq[-3] in rand:
•                 rand.remove(cpu_seq[-3])
•
•             temp = process_info[cpu_seq[-3]]
•             state = process_state[cpu_seq[-3]]
•             if state < len(temp):
•                 if list(temp[state])[0] == 'I':
•                     inp.append((cpu_seq[-3],int(list(temp[state])[1])))
•                 else:

```

```

        out.append((cpu_seq[-3],int(list(temp[state]))[1])))
        process_state[cpu_seq[-3]] += 1

    if cpu_busy:
        if cpu_seq[-2] != rand[0]:
            cpu_seq.append(current_time)
            cpu[cpu_seq[-3]] = cpu_free - current_time
            cpu_seq += [rand[0] , current_time]
            cpu_free = current_time + cpu[cpu_seq[-2]]

    elif not cpu_busy and len(cpu):
        cpu_busy = True
        cpu_seq += [rand[0] , current_time]
        cpu_free = current_time + cpu[rand[0]]

    if ip_free == current_time:
        ip_busy = False

        temp = process_info[input_seq[-3]]
        state = process_state[input_seq[-3]]
        if state < len(temp):
            if list(temp[state])[0] == 'C':
                rand += [input_seq[-3] for i in range(int( process_info[input_seq[-3]][1] ))]
                cpu[input_seq[-3]] = int(list(temp[state]))[1]
            else:
                out.append((input_seq[-3],int(list(temp[state]))[1])))
                process_state[input_seq[-3]] += 1

    if not ip_busy and len(inp):
        ip_busy = True
        input_seq += [inp[0][0] , current_time , current_time + inp[0][1]]
        ip_free = current_time + inp[0][1]
        del inp[0]

    if op_free == current_time:
        op_busy = False

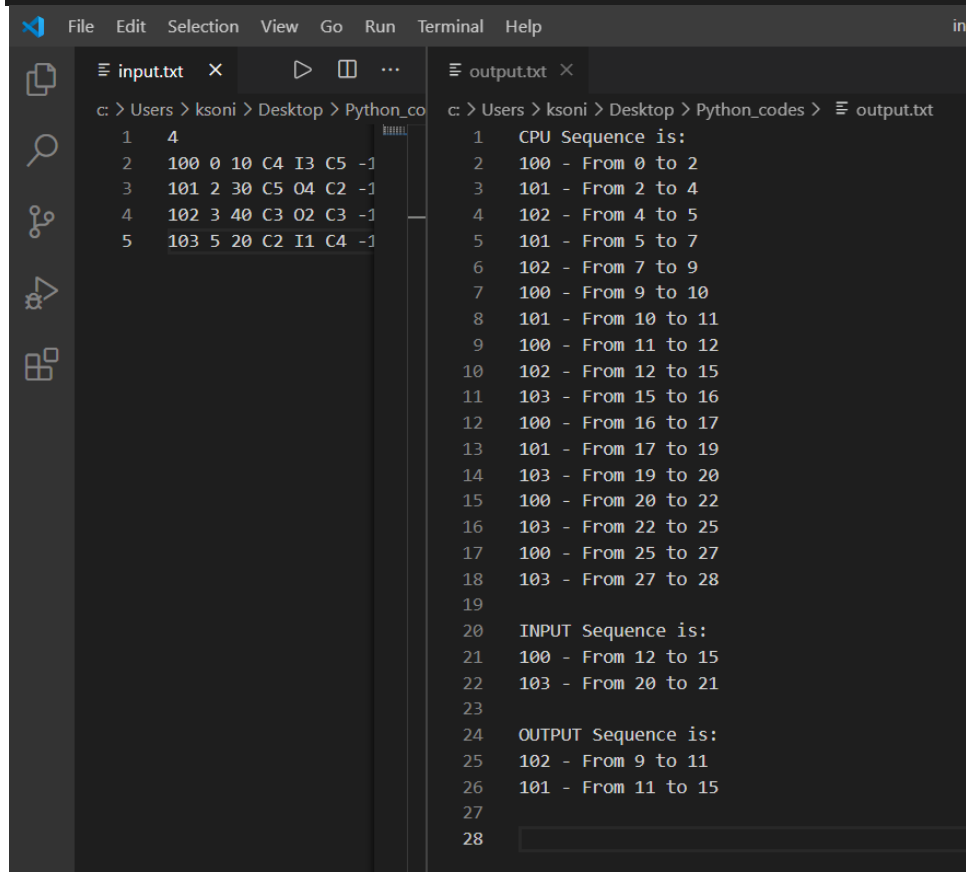
        temp = process_info[output_seq[-3]]
        state = process_state[output_seq[-3]]
        if state < len(temp):
            if list(temp[state])[0] == 'C':
                rand += [output_seq[-3] for i in range(int( process_info[output_seq[-3]][1] ))]
                cpu[output_seq[-3]] = int(list(temp[state]))[1]
            else:
                inp.append((output_seq[-3],int(list(temp[state]))[1])))
                process_state[output_seq[-3]] += 1

```

```

•
•     if not op_busy and len(out):
•         op_busy = True
•         output_seq += [out[0][0] , current_time , current_time + out[0][1]]
•         op_free = current_time + out[0][1]
•         del out[0]
•
•     current_time += 1
•     if current_time == 50:
•         break
•
• print("CPU Sequence is:")
• for i in range(0,len(cpu_seq),3):
•     print("{} - From {} to {}".format(cpu_seq[i],cpu_seq[i+1],cpu_seq[i+2]))
• print("")
•
• print("INPUT Sequence is:")
• for i in range(0,len(input_seq),3):
•     print("{} - From {} to {}".format(input_seq[i],input_seq[i+1],input_seq[i+2]))
• print("")
•
• print("OUTPUT Sequence is:")
• for i in range(0,len(output_seq),3):
•     print("{} - From {} to {}".format(output_seq[i],output_seq[i+1],output_seq[i+2]))
• print("")
•

```



```

c: > Users > ksoni > Desktop > Python_codes > input.txt
1 4
2 100 0 10 C4 I3 C5 -1
3 101 2 30 C5 04 C2 -1
4 102 3 40 C3 02 C3 -1
5 103 5 20 C2 I1 C4 -1

c: > Users > ksoni > Desktop > Python_codes > output.txt
1 CPU Sequence is:
2 100 - From 0 to 2
3 101 - From 2 to 4
4 102 - From 4 to 5
5 101 - From 5 to 7
6 102 - From 7 to 9
7 100 - From 9 to 10
8 101 - From 10 to 11
9 100 - From 11 to 12
10 102 - From 12 to 15
11 103 - From 15 to 16
12 100 - From 16 to 17
13 101 - From 17 to 19
14 103 - From 19 to 20
15 100 - From 20 to 22
16 103 - From 22 to 25
17 100 - From 25 to 27
18 103 - From 27 to 28
19
20 INPUT Sequence is:
21 100 - From 12 to 15
22 103 - From 20 to 21
23
24 OUTPUT Sequence is:
25 102 - From 9 to 11
26 101 - From 11 to 15
27
28

```