EXPERIMENT NO. 2

AIM: Write a program to implement SJF scheduling algorithm for process management.

RESOURCES REQUIRED:

H/W Requirements: P-IV and above, Ram 128 MB, Printer, Internet Connection.

S/W Requirements: Python compiler

THEORY:

Shortest Job First (SJF):

Shortest Job First (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN is a non pre-emptive algorithm.

- Shortest Job First has the advantage of having minimum average waiting time among all scheduling algorithms.
- It is a Greedy Algorithm.
- It may cause starvation if shorter processes keep coming. This problem can be solved using concept of aging.
- It is practically infeasible as Operating System may not know burst time and therefore may not sort them. While it is not possible to predict execution time, several methods can be used as a weighted average of previous execution times. SJF can be used in specialized environments where accurate estimates of running time are available.

CONCLUSION: Hence, we have implemented a program on SJF scheduling algorithm on process management.

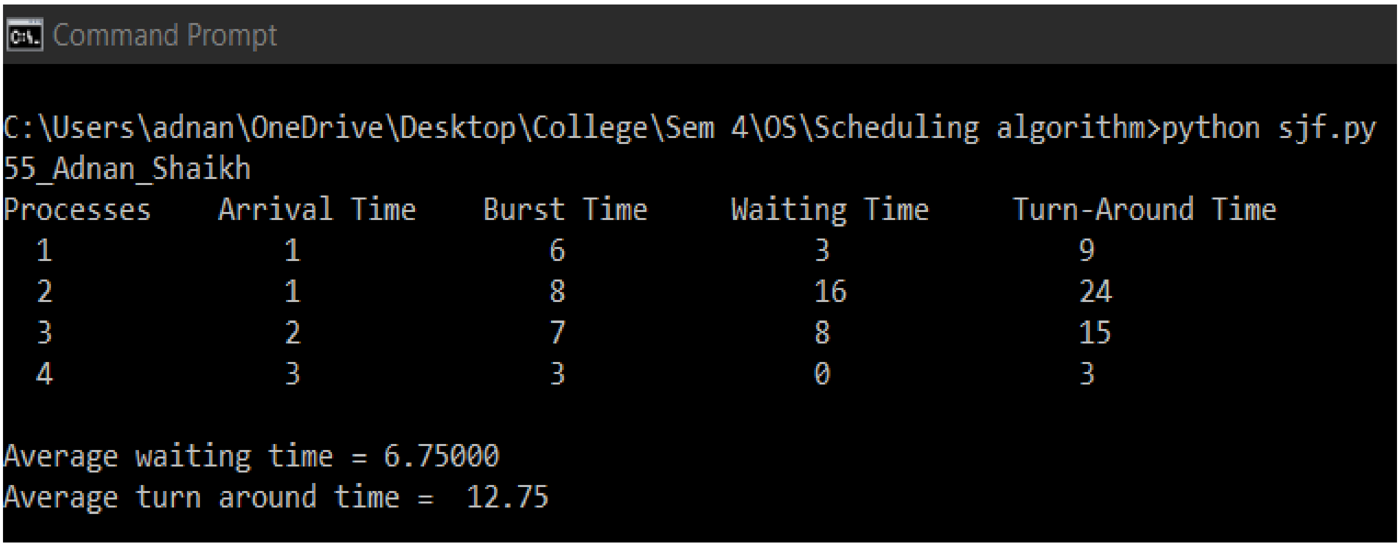
CODE:

```
from prettytable import PrettyTable


def sjf():

    processes = [[1, 6, 1], [2, 8, 1], [3, 7, 2], [4, 3, 3]]
#[processid,burst_time,arrival_time]
```

```
n = len(processes)
rt = [0] * n
wt = [0] * n
tat = [0] * n


for i in range(n):
    rt[i] = processes[i][1]
complete = 0
t = 0
minm = 999999999
short = 0
check = False


while (complete != n):

    for j in range(n):
        if ((processes[j][2] <= t) and (rt[j] < minm) and rt[j] > 0):
            minm = rt[j]
            short = j
            check = True
    if (check == False):
        t += 1
        continue
```

```
rt[short] -= 1



minm = rt[short]
if (minm == 0):
    minm = 999999999



if (rt[short] == 0):


    complete += 1
    check = False



    fint = t + 1



    wt[short] = (fint - processes[short][1] - processes[short][2])


    if (wt[short] < 0):
        wt[short] = 0


t += 1
for i in range(n):
    tat[i] = processes[i][1] + wt[i]
    print("Processes    Arrival Time    Burst Time     Waiting",
                "Time     Turn-Around Time")
    total_wt = 0
    total_tat = 0
    for i in range(n):

        total_wt = total_wt + wt[i]
        total_tat = total_tat + tat[i]
        print(" ", processes[i][0], "\t\t",
                    processes[i][2], "\t\t",
                    processes[i][1], "\t\t",
                    wt[i], "\t\t", tat[i])

    print("\nAverage waiting time = %.5f "%(total_wt /n) )
    print("Average turn around time = ", total_tat / n)

if __name__ == "__main__":
    print("55_Adnan_Shaikh")
    sjf()
```

OUTPUT:



```
C:\Users\adnan\OneDrive\Desktop\College\Sem 4\OS\Scheduling algorithm>python sjf.py
55_Adnan_Shaikh
Processes    Arrival Time    Burst Time     Waiting Time     Turn-Around Time
  1             1               6              3                9
  2             1               8              16               24
  3             2               7              8                15
  4             3               3              0                3

Average waiting time = 6.75000
Average turn around time =  12.75
```