OPERATING SYSTEM LAB 6

Roll No.: K041	Name: Anish Sudhan Nair
Batch No.: A2/K2	Date: 04/02/2022

Aim: To familiarise and implement the shortest remaining time first algorithm

1. Example 1

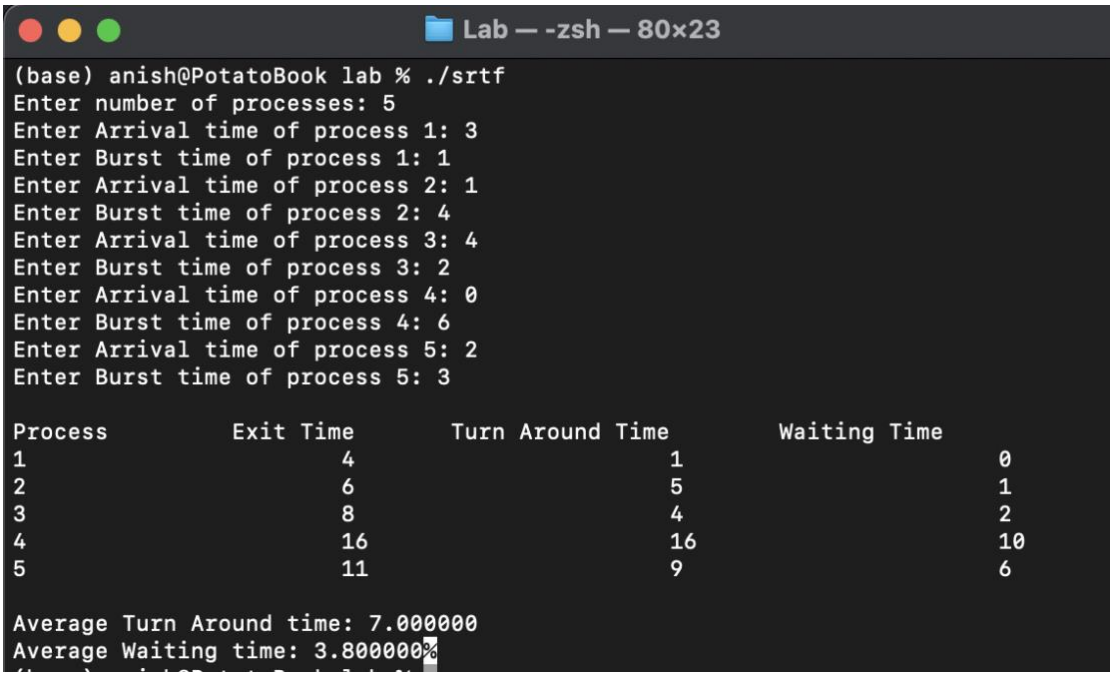
Example-01:

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

Output:

SRTF pre emptive done



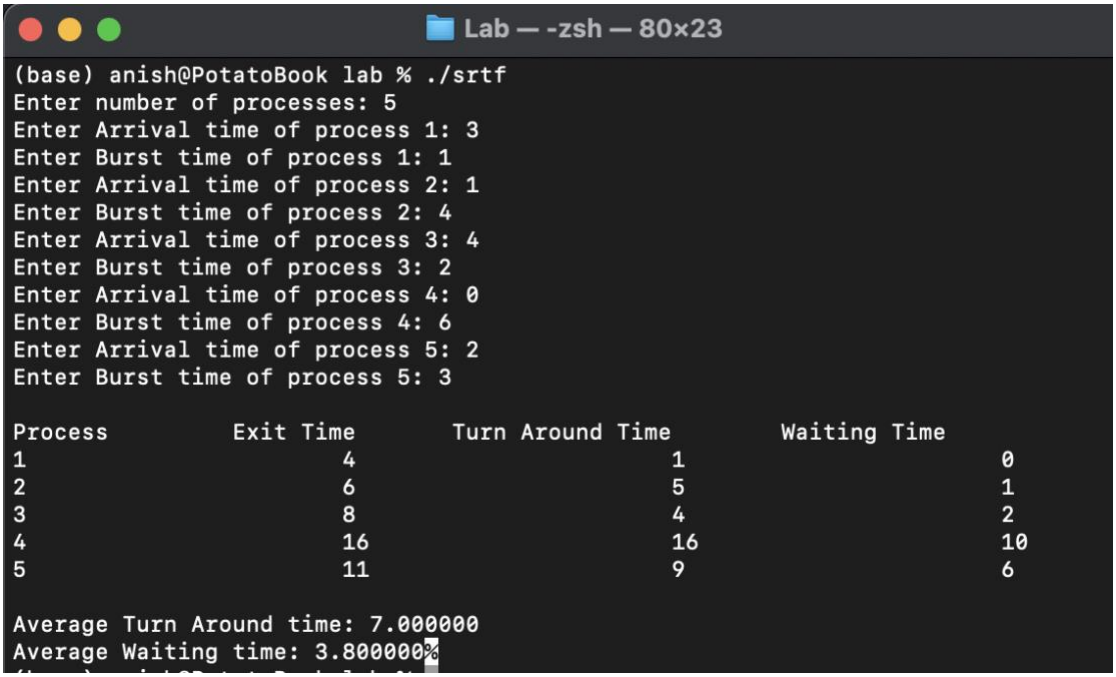
2. Example 2

Example-02:

Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

Output:



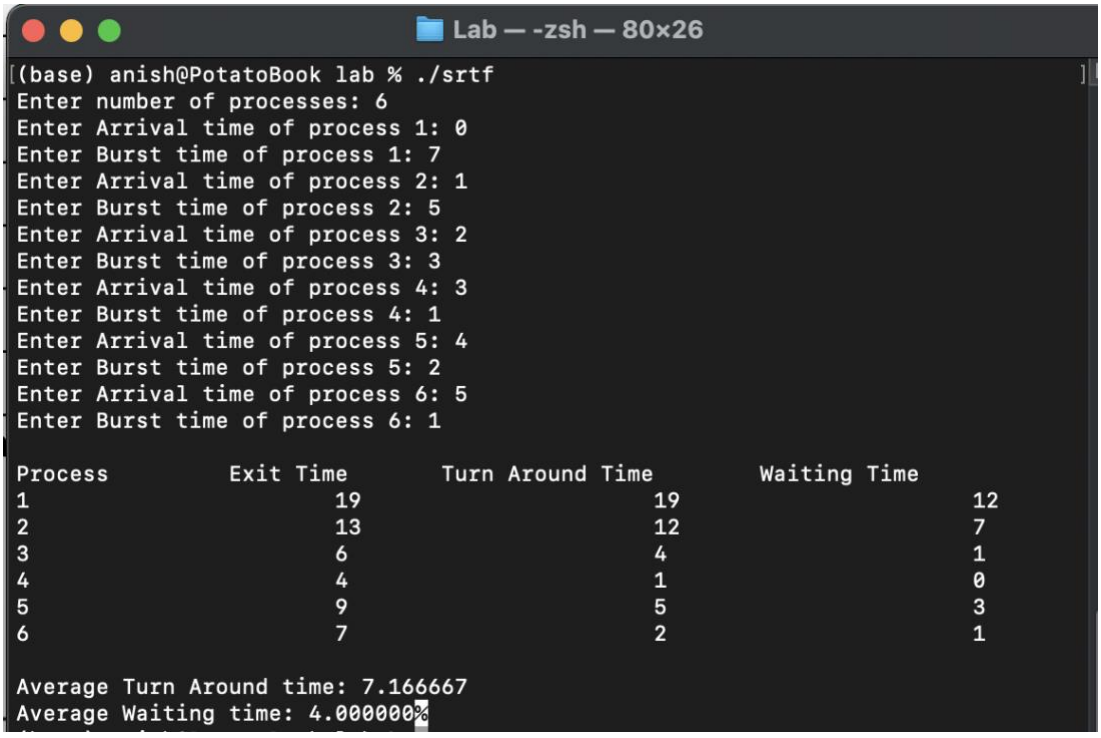
3. Example 3

Example-03:

Consider the set of 6 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	7
P2	1	5
P3	2	3
P4	3	1
P5	4	2
P6	5	1

Output:



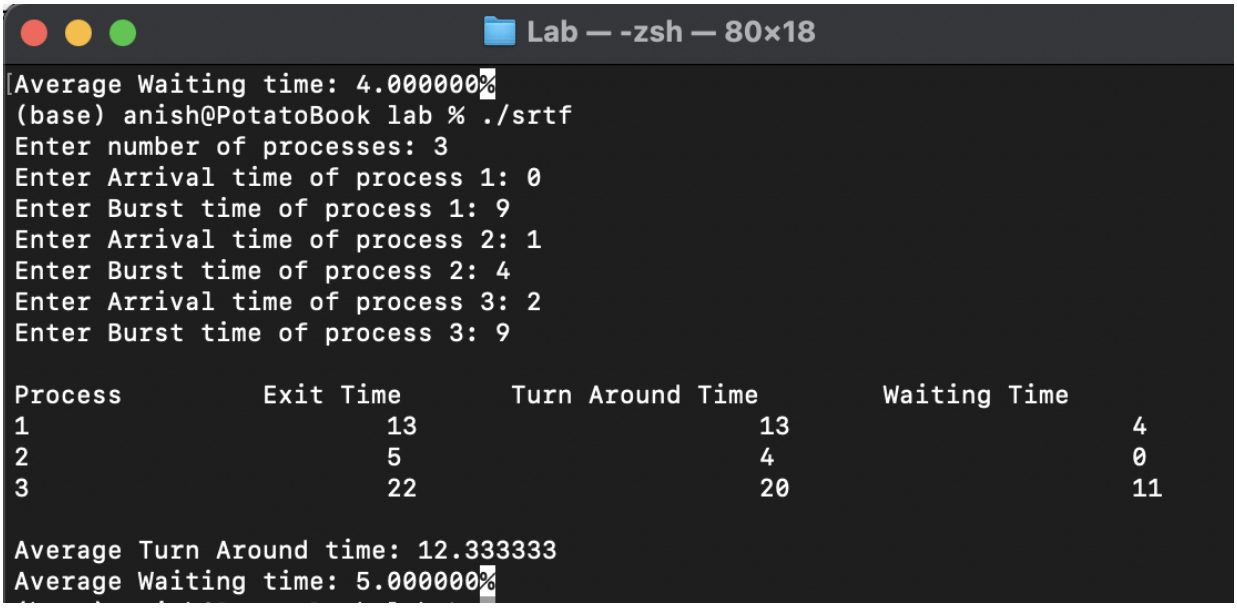
4. Example 4

Example -04:

Consider the set of 3 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	9
P2	1	4
P3	2	9

Output:



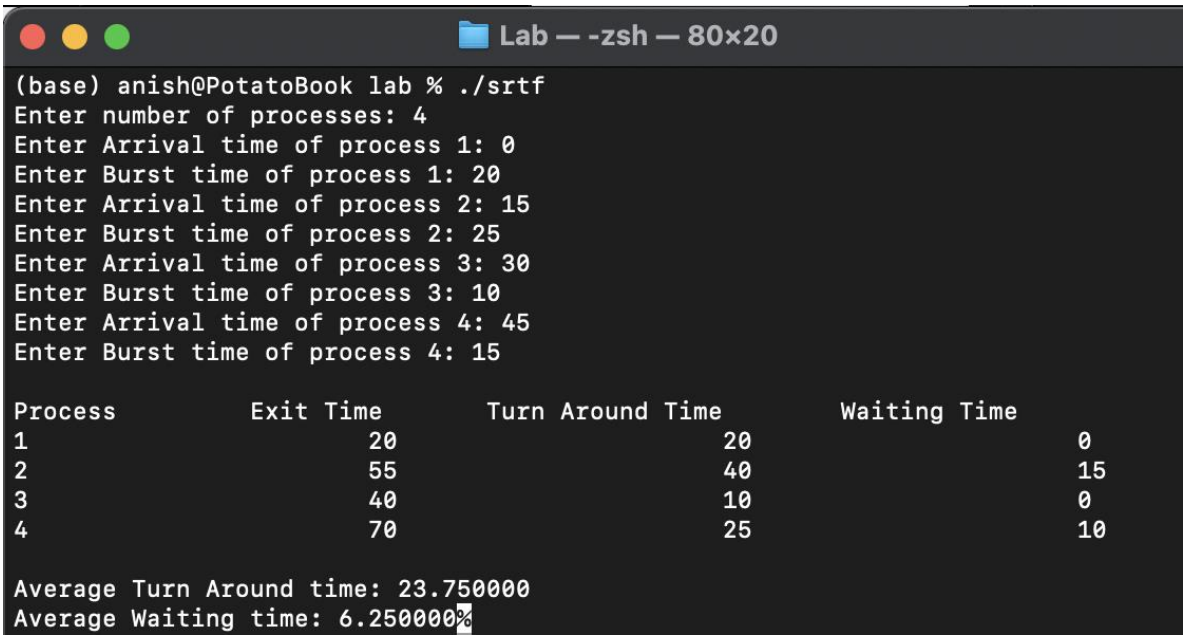
5. Example 5

Example-05:

Consider the set of 4 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	20
P2	15	25
P3	30	10
P4	45	15

Output:



Code:

```c
#include <stdio.h>
#include <stdlib.h>
char array0[16];
int num=0;
int Total,sumTOT=0,sumWT=0;

struct processes{
    int process_id;
    int arrivalTime;
    int burstTime;
    int exitTime;
    int turnAroundTime;
    int waitingTime;
    int ogBurstTime;
}p1,p2,p3,p4,p5,process[10];

int OGBT(){
    for(int i=0;i<Total;i++){
        process[i].ogBurstTime=process[i].burstTime;
    }
    return 0;
}

int funct(int array1[Total],int i)
{
    for(int u=0;u<Total;u++){
            if(process[array1[u]].burstTime>0){
                process[array1[u]].burstTime--;
                if(process[array1[u]].burstTime==0){
                    process[array1[u]].exitTime=i+1;
                }
                return array1;
            }
            }
            return array1;
}

int main()
{
    printf("Enter number of processes: ");
    scanf("%d",&Total);
    for(int i=0;i<Total;i++)
    {
        printf("Enter Arrival time of process %d: ",i+1);
        scanf("%d",&process[i].arrivalTime);
        printf("Enter Burst time of process %d: ",i+1);
        scanf("%d",&process[i].burstTime);
        process[i].process_id=i+1;
    }

    OGBT();

        for(int i=0;i<6946;i++){
            int array1[Total],z=0;
            for(int a=0;a<Total;a++){
                if(process[a].arrivalTime<=i&&process[a].burstTime>0){
                    array1[z]=a;
                    z++;
                }
```

```
            for(int j=0;j<z;j++){
                for(int k=j+1;k<z;k++){
                    if (process[array1[j]].burstTime > process[array1[k]].burstTime)
                    {
                    int w;
                    w =  array1[j];
                    array1[j]=array1[k];
                    array1[k]=w;

                    }

                }
            }
            }
            funct(array1,i);

            array0[i]=process[array1[0]].process_id;
            num=num+1;
            if(process[0].burstTime+process[1].burstTime+process[2].burstTime+process[3].burstTime+process[4].burstTime==0){
                break;
            }
        }
        printf("\n");
        printf("Process \tExit Time \tTurn Around Time \tWaiting Time\n");
        for(int z=0;z<Total;z++)
        {

            process[z].turnAroundTime=process[z].exitTime-process[z].arrivalTime;
            process[z].waitingTime=process[z].turnAroundTime-process[z].ogBurstTime;
            sumTOT+=process[z].turnAroundTime;
            sumWT+=process[z].waitingTime;
            printf("%d\t\t\t",process[z].process_id);
            printf("%d\t\t\t",process[z].exitTime);
            printf("%d\t\t\t",process[z].turnAroundTime);
            printf("%d\n",process[z].waitingTime);
        }
        printf("\nAverage Turn Around time: %f",((float)sumTOT/Total));
        printf("\nAverage Waiting time: %f",((float)sumWT/Total));

    }
```

CONCLUSION:

In this lab, we were to implement and demonstrate the working of the shortest remaining time first algorithm. By actually coding the algorithm, it helped to reinforce the working of this process and the manner in which it schedules the processes in a CPU.