Name: Shawn Louis

Roll No: 31

Batch: B

Topic:	To write a program to implement CPU scheduling algorithm pre-emptive & non pre-emptive SJF
Prerequisite:	Basic knowledge of using the linux terminal and system calls, and C language.
Mapping With COs:	CSL404.4
Theory:	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 the 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 to estimate the execution time for a job, such as a weighted average of previous execution times. SJF can be used in specialized environments where accurate estimates of running time are available.
Objective:	• To understand & analyse SJF algorithm • To implement SJFalgorithm • calculate average waititng time &turn around time

Program Code:	Non-Premptive:
	#include<stdio.h>
	int main()
	{
	int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
	float avg_wt,avg_tat;
	printf("Enter number of process:");
	scanf("%d",&n);

```
 printf("\nEnter Burst Time:\n");
 for(i=0;i<n;i++)
 {
    printf("p%d:",i+1);
    scanf("%d",&bt[i]);
    p[i]=i+1;
 }

//sorting of burst times
 for(i=0;i<n;i++)
 {
    pos=i;
    for(j=i+1;j<n;j++)
    {
       if(bt[j]<bt[pos])
          pos=j;
    }

    temp=bt[i];
    bt[i]=bt[pos];
    bt[pos]=temp;

    temp=p[i];
    p[i]=p[pos];
    p[pos]=temp;
 }

 wt[0]=0;
```

```
   for(i=1;i<n;i++)
   {
      wt[i]=0;
      for(j=0;j<i;j++)
         wt[i]+=bt[j];

      total+=wt[i];
   }

   avg_wt=(float)total/n;
   total=0;

   printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
   for(i=0;i<n;i++)
   {
      tat[i]=bt[i]+wt[i];
      total+=tat[i];
      printf("\np%d\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
   }

   avg_tat=(float)total/n;
   printf("\n\nAverage Waiting Time=%f",avg_wt);
   printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

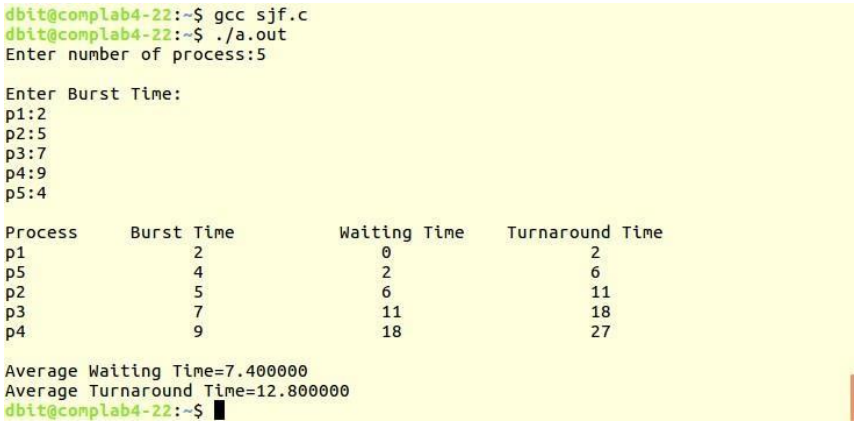
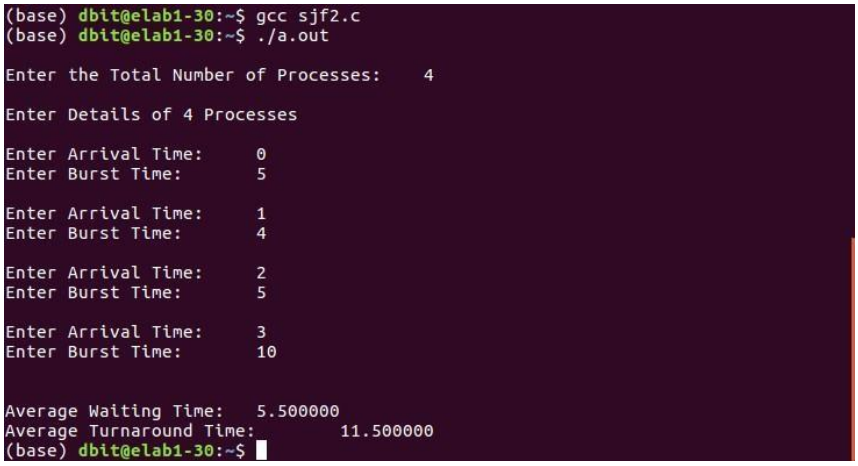
Premptive:

```
#include <stdio.h>

int main()
```

```
{
    int arrival_time[10], burst_time[10], temp[10];
    int i, smallest, count = 0, time, limit;
    double wait_time = 0, turnaround_time = 0, end;
    float average_waiting_time, average_turnaround_time;
    printf("\nEnter the Total Number of Processes:\t");
    scanf("%d", &limit);
    printf("\nEnter Details of %d Processes\n", limit);
    for(i = 0; i < limit; i++)
    {
        printf("\nEnter Arrival Time:\t");
        scanf("%d", &arrival_time[i]);
        printf("Enter Burst Time:\t");
        scanf("%d", &burst_time[i]);
        temp[i] = burst_time[i];
    }
    burst_time[9] = 9999;
    for(time = 0; count != limit; time++)
    {
        smallest = 9;
        for(i = 0; i < limit; i++)
        {
            if(arrival_time[i] <= time && burst_time[i] <
burst_time[smallest] && burst_time[i] > 0)
            {
                smallest = i;
            }
        }
        burst_time[smallest]--;
        if(burst_time[smallest] == 0)
        {
```

```
            count++;
            end = time + 1;
            wait_time = wait_time + end - arrival_time[smallest] -
temp[smallest];
            turnaround_time = turnaround_time + end -
arrival_time[smallest];
        }
    }
    average_waiting_time = wait_time / limit;
    average_turnaround_time = turnaround_time / limit;
    printf("\n\nAverage Waiting Time:\t%lf\n",
average_waiting_time);
    printf("Average Turnaround Time:\t%lf\n",
average_turnaround_time);
    return 0;
}
```

Output Snapshot:	 
Outcome:	• Abilityto **implementandanalyze if** eventprocess schedulingalgorithms
Conclusion:	Students will learn to analyze and implement pre-emptive & non pre-emptive SJF algorithm.
References:	Reference document along with the assignment Internet facility is available to explore further .