**Name – Shri Prakash Yadav**

**Regno – 11811471**

**Sec – K18GE**

**Sub - operating system**

**Github Link -**

**Code: C Program**

**#include<stdio.h>**

**#include<conio.h>**

**int main()**

**{**

**int bt[10],p[10],at[10],n,temp,tt,i,j,wt[10],sum=0,total=0;**

**float avg;**

**printf("Enter total no of proces:\n");**

**scanf("%d",&n);**

**printf("Enter burst time for each process:-\n");**

**for(i=0;i<n;i++)**

**{**

**printf("Burst time of process P%d:\n",i);**

**scanf("%d",&bt[i]);**

**p[i]=i;**

**printf("Arrival time of process P%d\n",i);**

**scanf("%d",&at[i]);**

**}**

**for(i=0;i<n-1;i++)**

**{**

**for(j=i+1;j<n;j++)**

**{**

**if(bt[i]>bt[j])**

**{**

**temp=bt[i];**

**bt[i]=bt[j];**

**bt[j]=temp;**

**temp=p[i];**

**p[i]=p[j];**

**p[j]=temp;**

**}**

**}**

**}**

**wt[0]=0;**

**for(i=1;i<n;i++)**

**{**

**wt[i]=wt[i-1]+bt[i-1];**

**}**

**for(i=0;i<n;i++)**

**{**

**sum+=wt[i];**

**}**

**avg=(float)sum/n;**

**printf("Waiting time for each process:-\n");**

**for(i=0;i<n;i++)**

**{**

**printf("Waiting time for process P%d is %d sec.\n",p[i],wt[i]);**

**}**

**printf("Average waiting time is %f sec.\n",avg);**

**for(i=0;i<n;i++)**

**{**

**tt=bt[i]+(2\*n);**

**}**

**printf("total time %d",tt);**

**getch();**

**return 0;**

**}**

**Algorithm:**

1. Sort all the process according to the arrival time.
2. Then select that process which has minimum arrival time and minimum Burst time.
3. After completion of process make a pool of process which after till the completion of previous process and select that process among the pool which is having minimum Burst time.

**Complexity Of The Algorithm:**

The complexity of the shortest job first scheduling is O(n log n).

**Constraints**

* In SJF Scheduling, a process with high burst time may suffer starvation. Starvation is the process in which a process with higher burst time is kept on waiting and waiting, but is not allocated to the CPU. It’s prevented by **Aging.**
* Total execution time must be known beforehand of a process.

**Boundary Condition:**

Once resources (CPU Cycle) are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.

Process cannot be interrupted until it terminates itself or its time is up.

If a process with long burst time is running CPU, then later coming process with less CPU burst time may starve.

It does not have overheads.

The process is rigid.

No cost associated.

**Test Cases:**

**Case 1:**

|  |  |  |
| --- | --- | --- |
| Process Queue | Burst time | Arrival time |
| P1 | 6 | 2 |
| P2 | 2 | 4 |
| P3 | 7 | 2 |
| P4 | 2 | 0 |
| P5 | 4 | 3 |

Wait time

P1= 0

P3= 2

P4= 4

P0= 8

P2= 14

Average Waiting Time=5.6

