**SJF SCHEDULING ALGORITHM**

**Shortest job first** (**SJF**) also known as **Shortest job next** (**SJN**), is a scheduling algorithm that selects for execution the waiting process with the smallest execution time. SJF is a non-pre-emptive algorithm. Shortest Remaining Time First is a pre-emptive variant of SJF.

**ADVANTAGES**

* Maximum throughput.

Throughput means total number of processes executed per unit time. Shortest job first scheduling algorithm selects the waiting process with the smallest execution time. Thus, in SJF, shortest jobs are executed first making the CPU utilization maximum. So, maximum number of tasks are completed.

* Minimum waiting and turn around time as compared with other scheduling algorithms.

**DISADVANTAGES**

* It may cause **starvation** if shorter process keeps coming.

In case process with lower burst time appears before process with higher burst time then only the starvation may occur, since the algorithm will always choose the process with lowest Burst Time, the process with higher Burst Time will never be able to get the share of CPU.

**ALGORITHM**

1. Sort all the processes according to their arrival time.
2. Select the process with minimum arrival time as well as minimum burst time.
3. After completion of the process, select from the ready queue the process which has the minimum burst time.
4. If processes have same burst time length then FCFS ( First come First Serve ) scheduling algorithm used.
5. Repeat above processes until all processes have finished their execution.

**CODE**

#include<stdio.h>

int main()

{

            int at[10],bt[10],pr[10];

            int n,i,j,temp,time=0,count,over=0,sum\_wait=0,sum\_turnaround=0,start;

            float avgwait,avgturn;

            printf("Enter the number of processes\n");

            scanf("%d",&n);

            for(i=0;i<n;i++)

            {

                        printf("Enter the arrival time and execution time for process %d\n",i+1);

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

                        pr[i]=i+1;

            }

            for(i=0;i<n-1;i++)

            {

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

                        {

                                    if(at[i]>at[j])

                                    {

                                                temp=at[i];

                                                at[i]=at[j];

                                                at[j]=temp;

                                                temp=bt[i];

                                                bt[i]=bt[j];

                                                bt[j]=temp;

                                                temp=pr[i];

                                                pr[i]=pr[j];

                                                pr[j]=temp;

                                    }

                        }

            }

            printf("\n\nProcess\t|Arrival time\t|Execution time\t|Start time\t|End time\t|waiting                                     time\t|Turnaround time\n\n");

            while(over<n)

            {

                        count=0;

                        for(i=over;i<n;i++)

                        {

                                    if(at[i]<=time)

                                    count++;

                                    else

                                    break;

                        }

                        if(count>1){

                                    for(i=over;i<over+count-1;i++)

                                    {

                                                for(j=i+1;j<over+count;j++)

                                                {

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

                                                            {

                                                                        temp=at[i];

                                                                        at[i]=at[j];

                                                                        at[j]=temp;

                                                                        temp=bt[i];

                                                                        bt[i]=bt[j];

                                                                        bt[j]=temp;

                                                                        temp=pr[i];

                                                                        pr[i]=pr[j];

                                                                        pr[j]=temp;

                                                            }

                                                }

                                    }

                        }

                        start=time;

                        time+=bt[over];

                         printf("p[%d]\t|\t%d\t|\t%d\t|\t%d\t|\t%d\t|\t%d\t|\t%d\n",pr[over],

                                    at[over],bt[over],start,time,time-at[over]-bt[over],time-at[over]);

                        sum\_wait+=time-at[over]-bt[over];

                        sum\_turnaround+=time-at[over];

                        over++;

            }

            avgwait=(float)sum\_wait/(float)n;

            avgturn=(float)sum\_turnaround/(float)n;

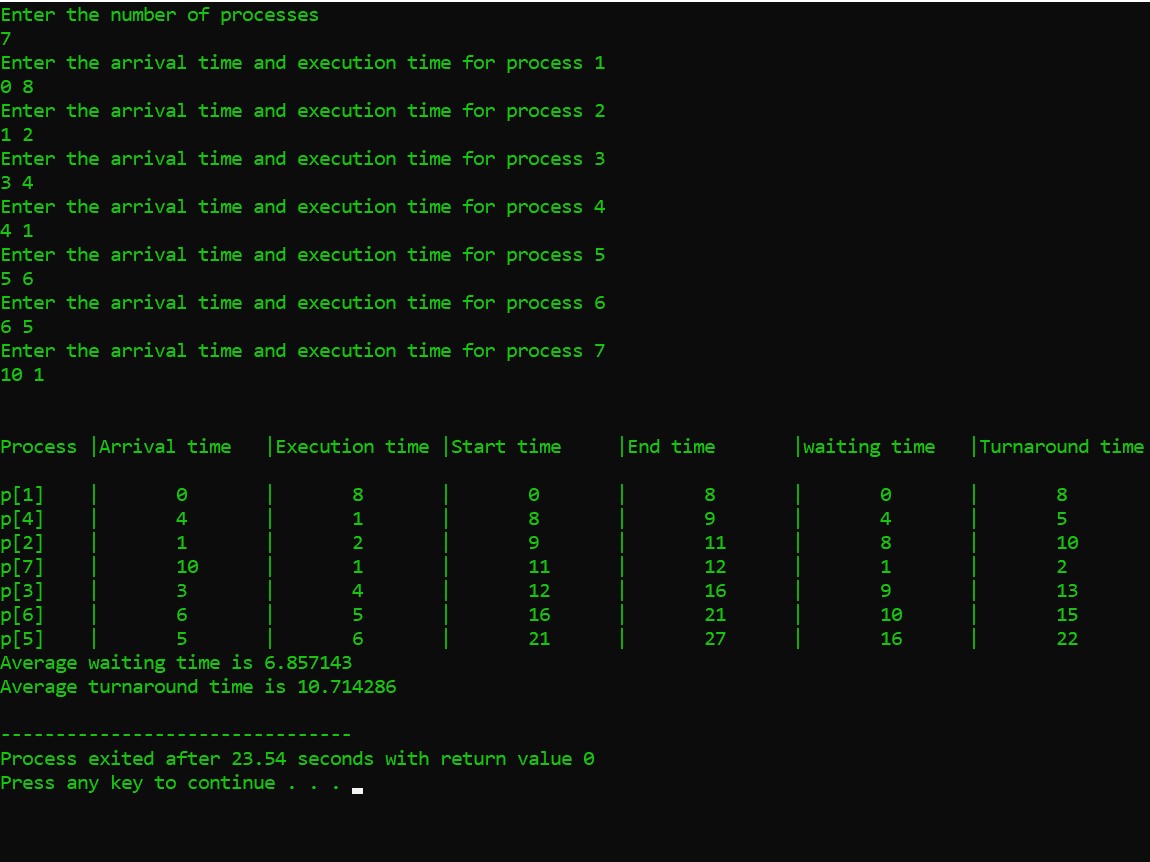
            printf("Average waiting time is %f\n",avgwait);

            printf("Average turnaround time is %f\n",avgturn);

            return 0;

}

**SAMPLE OUTPUT:**

****