OPERATING SYSTEM

CSE-316

PROJECT TITLE:

SHORTEST JOB FIRST SCHEDULING (SJF)

LOVELY PROFESSIONAL UNIVERSITY

NAME: PRINCE KUMAR
SECTION: K21WY-G2
REGI NO: 12112403
ROLL NO: RK21WYA02

SUBMITTED TO:
MS.RICHA SHARMA

SOURCE CODE

```
#include<stdio.h>
int main()
{
    //n    number of processes
    //bt   burst time
    //p    process
    //at   arrays to store the arrival time
    //wt   waiting time
    //tat  turnaround time
    //rt   remaining time
    //ct   completion time
    // variables for loop counters i and j and a temporary variable temp
for sorting

    int n, bt[20],P[20], at[20], wt[20], tat[20], rt[20], ct[20], i, j,
temp;
    float avg_wt, avg_tat;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("\nEnter the arrival time & burst time for each process:\n");
    for(i=0; i<n; i++)
    {
        printf("P%d: ", i+1);
        P[i]=i+1;
        scanf("%d %d", &at[i], &bt[i]);
        rt[i]=bt[i];
    }
```

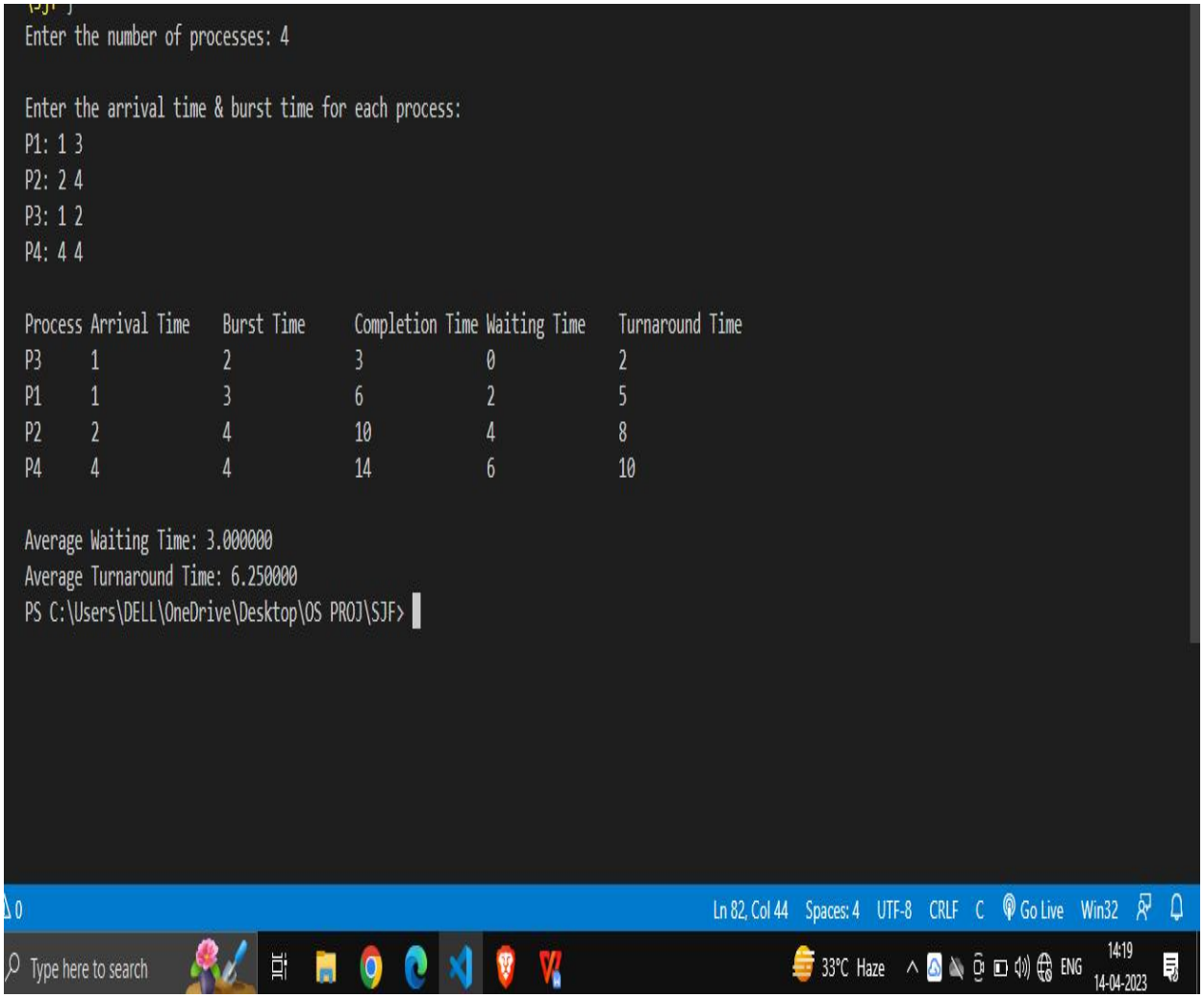
```
// sorting processes based on arrival time using selection sort
    for(i=0; i<n; i++)
    {
        for(j=i+1; j<n; j++)
        {
            if(at[i]>=at[j])
            {   //sorting arival time
                temp=at[i];
                at[i]=at[j];
                at[j]=temp;
               //sorting bt
                temp=bt[i];
                bt[i]=bt[j];
```

```
       bt[j]=temp;
              //sorting rt remaining time
               temp=rt[i];
               rt[i]=rt[j];
               rt[j]=temp;
              //sorting process
               temp=P[i];
               P[i]=P[j];
               P[j]=temp;
           }
        }
    }
```

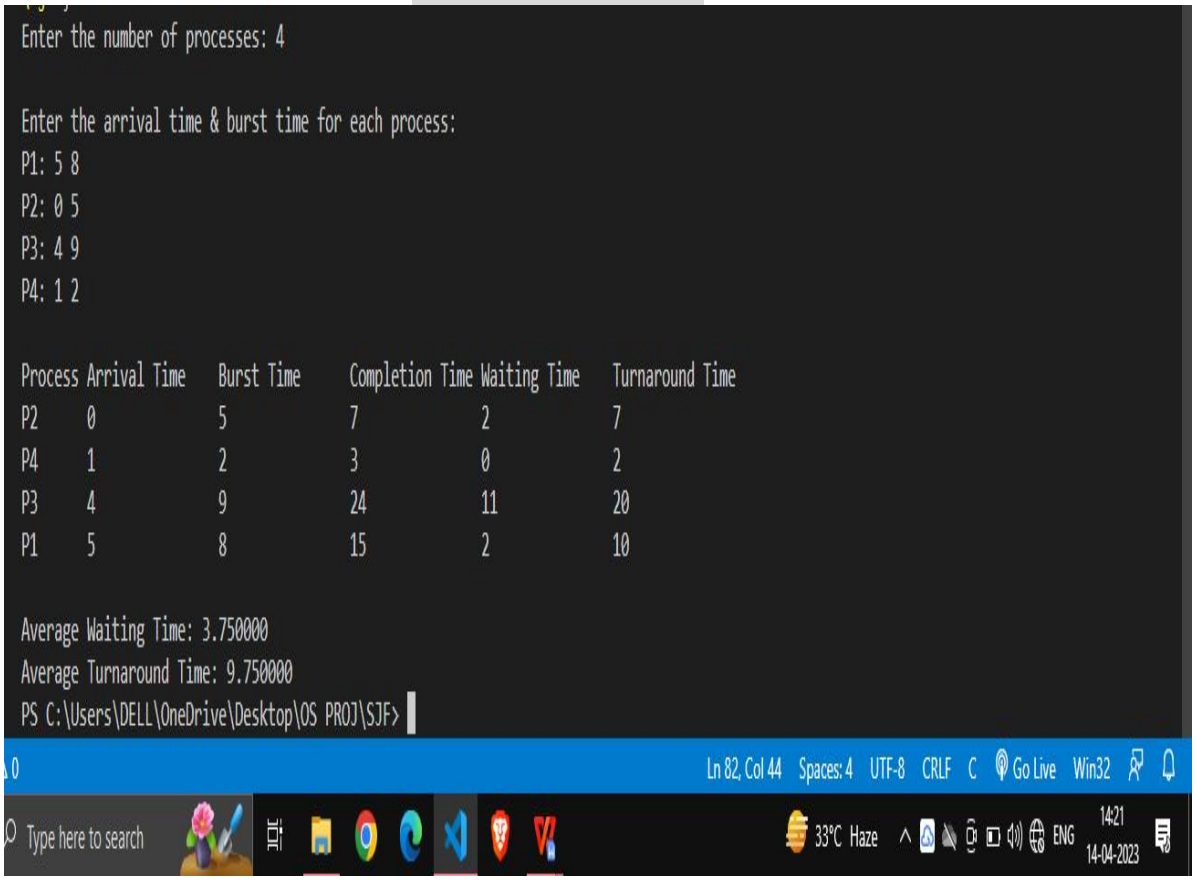
```
//algorithm for sjf
    int time=at[0], done=0, min_bt, k;
    while(done!=n)
    {
        min_bt=9999; // assuming maximum burst time to be 9999
        k=-1;
        for(i=0; i<n; i++)
        {
            if(rt[i]>0 && at[i]<=time && rt[i]<min_bt)
            {
                min_bt=rt[i];
                k=i;
            }
        }
        if(k==-1) // no process available to execute
        {
            time++;
            continue;
        }
        // executing the process
        rt[k]--;
        time++;
        if(rt[k]==0) // process execution completed
        {
            done++;
            ct[k]=time;
            tat[k]=ct[k]-at[k];
            wt[k]=tat[k]-bt[k];
            if(wt[k]<0) wt[k]=0;
        }
    }
```

```
 // calculating average waiting time and
    //average turnaround time and printing it
    avg_wt=0;
    avg_tat=0;
    printf("\nProcess\tArrival Time\tBurst Time\tCompletion
Time\tWaiting Time\tTurnaround Time\n");
    for(i=0; i<n; i++)
    {
        printf("P%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", P[i], at[i], bt[i],
ct[i], wt[i], tat[i]);
        avg_wt+=wt[i];
        avg_tat+=tat[i];
    }
    avg_wt/=n;
    avg_tat/=n;
    printf("\nAverage Waiting Time: %.2f", avg_wt);
    printf("\nAverage Turnaround Time: %.2f\n", avg_tat);
    return 0;
}
```

OUTPUT1:


Enter the number of processes: 4
Enter the arrival time & burst time for each process:
P1: 1 3
P2: 2 4
P3: 1 2
P4: 4 4
Process Arrival Time Burst Time Completion Time Waiting Time Turnaround Time
P3 1 2 3 0 2
P1 1 3 6 2 5
P2 2 4 10 4 8
P4 4 4 14 6 10
Average Waiting Time: 3.000000
Average Turnaround Time: 6.250000
PS C:\Users\DELL\OneDrive\Desktop\OS PROJ\SJF>

OUTPUT2:


Enter the number of processes: 4
Enter the arrival time & burst time for each process:
P1: 5 8
P2: 0 5
P3: 4 9
P4: 1 2
Process Arrival Time Burst Time Completion Time Waiting Time Turnaround Time
P2 0 5 7 2 7
P4 1 2 3 0 2
P3 4 9 24 11 20
P1 5 8 15 2 10
Average Waiting Time: 3.750000
Average Turnaround Time: 9.750000
PS C:\Users\DELL\OneDrive\Desktop\OS PROJ\SJF>
Ln 82, Col 44 Spaces: 4 UTF-8 CRLF C Go Live Win32
Type here to search
33°C Haze
ENG
14:21
14-04-2023