Student Name: Utkarsh Dubey

Student E-mail Address: davyutkarsh22@gmail.com

GitHub Link: https://github.com/utkarsh2223/Utkarsh-Project/tree/110d037f388ad596916310179441b60453c042fd

Code:

```
#include<stdio.h>

#include<stdlib.h>

void main()

{

int burst[20],p[20],waiting[20],turnat[20], n ,total=0,pos,temp;

int i,j;

float avg_w,avg_ta;

printf("Enter number of process:");

scanf("%d",&n);

printf("\nEnter Estimated Time:\n");


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

{

printf("p%d:",i+1);

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

p[i]=i+1;
```

```
}


for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(burst[j]<burst[pos])
pos=j;
}
temp=burst[i];
burst[i]=burst[pos];
burst[pos]=temp;
temp=p[i];
p[i]=p[pos];
p[pos]=temp;
}
waiting[0]=0;
for(i=1;i<n;i++)
{
waiting[i]=0;
```

```
for(j=0;j<i;j++)

waiting[i]+=burst[j];

total+=waiting[i];

}

avg_w=(float)total/n;

total=0;

printf("\nProcess\t Estimated Time \tWaiting Time\tTurnaround Time");

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

{

turnat[i]=burst[i]+waiting[i];

total+=turnat[i];

printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],burst[i],waiting[i],turnat[i]);

}

avg_ta=(float)total/n; //average turnaround time

printf("\n\nAverage Waiting Time=%f",avg_w);

printf("\nAverage Turnaround Time=%f\n",avg_ta);

}
```

**1.Explain the problem in terms of operating system concept ? (Max200word) Description:**

Ans. Process **Scheduling** is an **OS** task that **schedules** processes of different states like ready, waiting, and running. Process **scheduling** allows **OS** to allocate a time interval of CPU execution for each process. Another important reason for using a process **scheduling** system is that it keeps the CPU busy all the time.

**Arrival Time:** Time at which the process arrives in the ready queue.

**Completion Time:** Time at which process completes its execution.

**Burst Time:** Time required by a process for CPU execution.

**Turn Around Time:** Time Difference between completion time and arrival time.

Turn Around Time = Completion Time – Arrival Time

**Waiting Time(W.T):** Time Difference between turn around time and burst time.
Waiting Time = Turn Around Time – Burst Time

The question states that we have to use shortest job non pre-emptive scheduling . Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next.

We have to give four processes and make them run according to the SJF approach. Then our implemented code will compute its waiting time, turn-around time etc.

**2.Write the algorithm for proposed solution of the assigned problem. Algorithm.**

**Ans:**

## Algorithm -

Step 1 - Start.

Step 2 - Declare variable i, j.

Step 3 - Declare variable burst[20], p[20], waiting[20], turnat[20], n, total = 0, pos, temp;

Step 4 - Declare variable avg_w, avg_ta;

Step 5: for(i = 0; i < n; i++)
{ scanf("%d", &burst[i]);
p[i] = i+1; }

Step 6 - for(i=0; i<n; i++)
{ pos = i;
for(j = i+1; j<n; j++)
{ if burst[j] < burst[pos])
pos = j;
}
temp = burst[i];
burst[i] = burst[pos];
burst[pos] = temp;
temp = p[i];
p[i] = p[pos];
p[pos] = temp;
}

Step 7 - waiting[i] = 0;

Step 8 - for(i = 1; i<n; i++)
{ waiting[i] = 0;

Date
Page

~~Step 8~~ for(j=0; j<i; j++)
waiting[i] += burst[j];
total += waiting[i];
}

Step 9 - avg_w = (float) total/n;
total = 0;

Step 10 - for(i=0; i<n; i++)
{ turnat[i] = burst[i] + waiting[i];
total += turnat[i];
print( Process, burst time, waiting Time,
& turn around time)
}

Step 11 - avg_ta = (float) total/n;
print(avg_w);
print(avg_ta);

**Q3. Calculate complexity of algorithm.**

Ans. 1. Overall Time complexity : O(n*2)

2. OverallSpaceComplexity: O(n)

**Q4.Explain all the constrints given in the problem. Attach the code snippet.**

Ans. * The code implements the priority scheduling algorithm in which the priority of each process changes after each process gets the CPU.

*One of the constraints is that the code assumes that all the process arrives at the same time

*There are two options to enter the burst time for each process. One is to enter the burst time manually and another is to generate it randomly.

* Code snippet attached with question 7

**Q5. If you haveimplemented any additional algorithm to support the solution, explain the need and usage of the same. Description:**

Ans. Yes I have taken the help of selection sort. As I have to sort the burst time in ascending order, so I have implemented it in my program.

**Q6. Explain the boundary conditions of the implemented code. Description:**

Ans.* Boundary condition for Arrival time:0

* The arrival time for all process is Zero, because we assume that all the processes arrive at the same time.

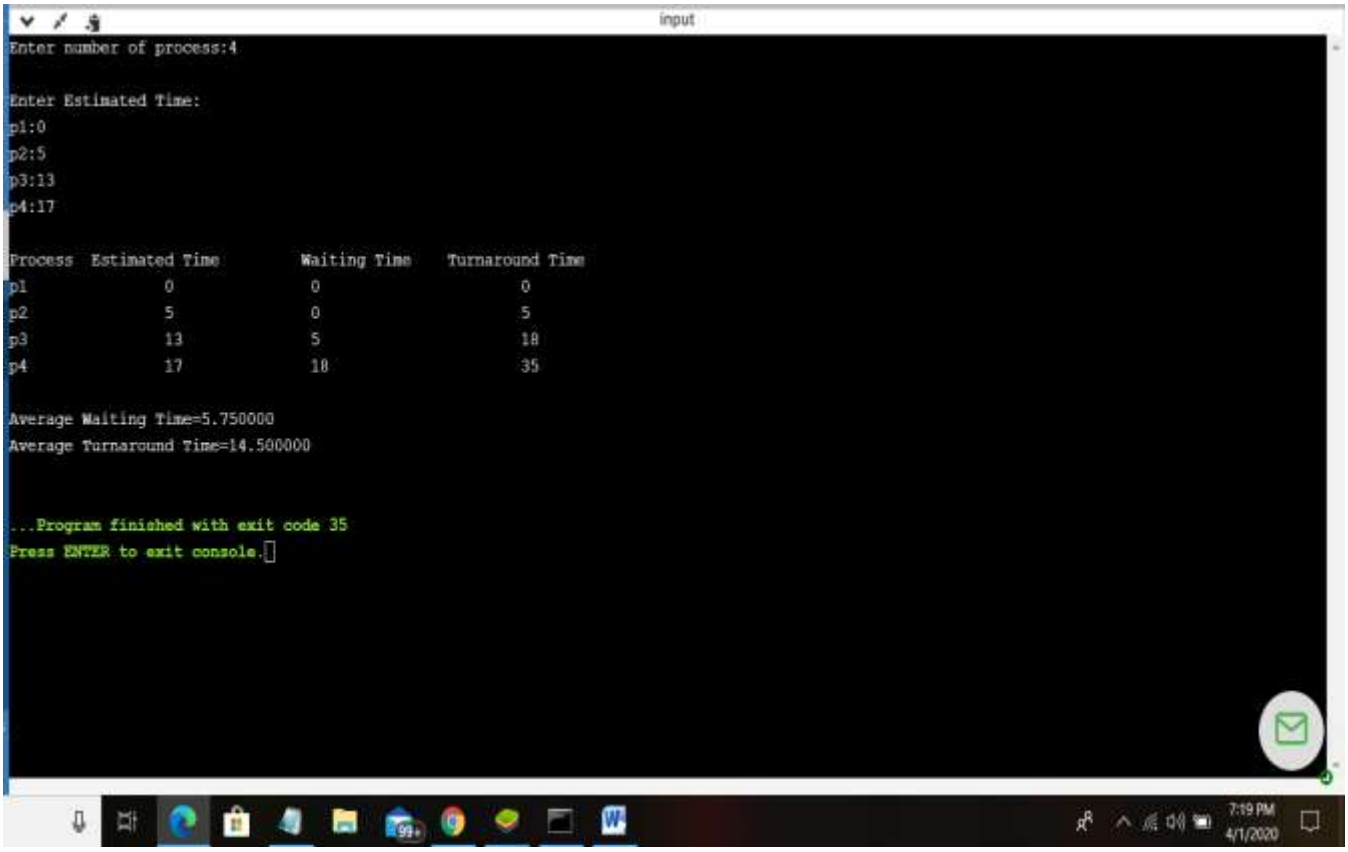
* Boundary condition for Burst time: 0 to + infinity

* The value of burst time cannot be negative because its not possible , whereas the minimum value can be 0, and maximum value could any thing greater than 0 as per the requirement of the process.

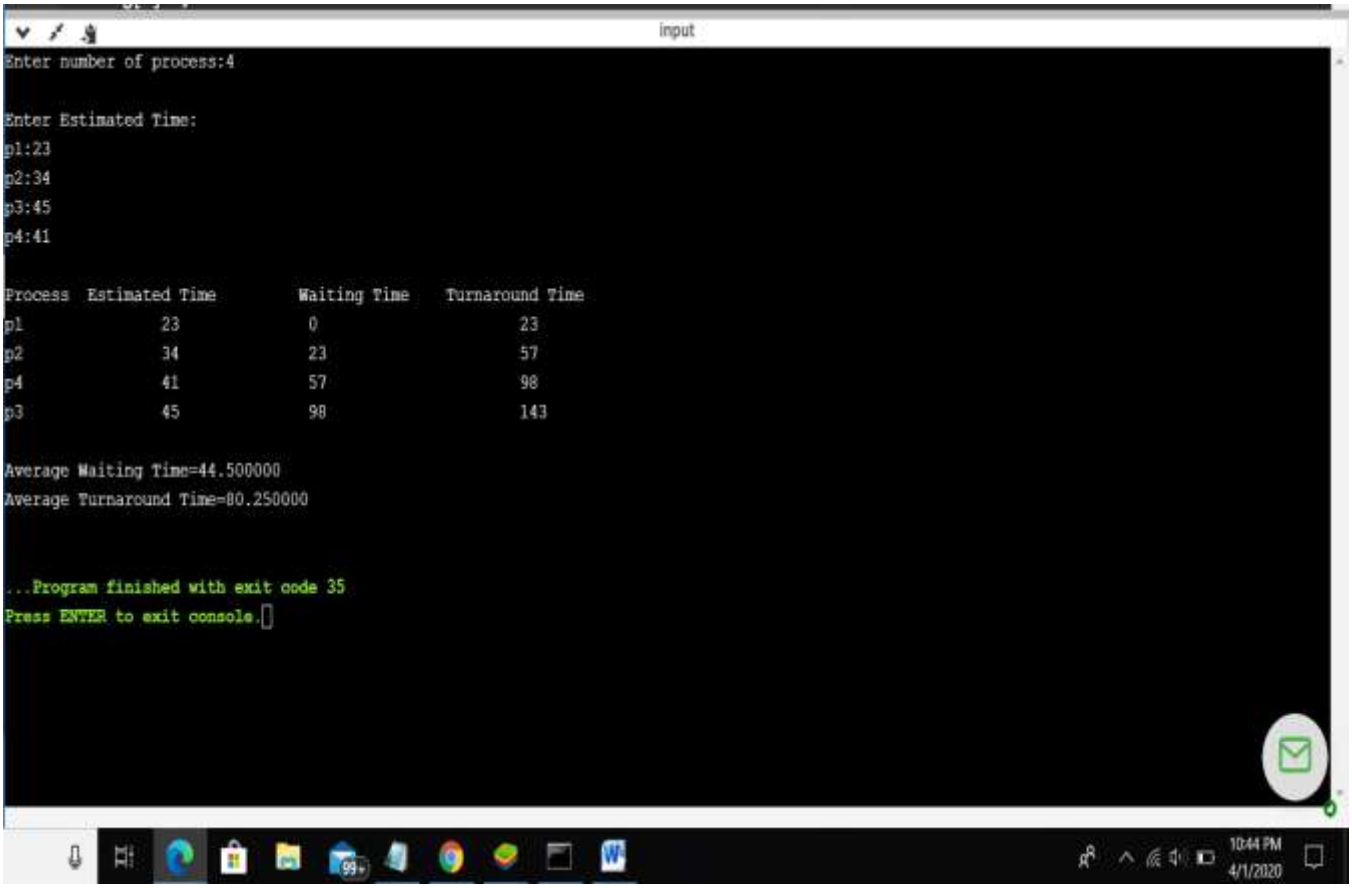
* Boundary condition for priority : 0 to 99

**Q7.Explain all the test cases applied on the solution of assigned problem. Description:**

Ans. **Given Input:**




**Randomly Generated Input:**




Q8. Have you made minimum 5 revisions of solution on Git Hub ? Git Hub Link:

Ans. Yes

GitHub Link: https://github.com/utkarsh2223/Utkarsh-Project/tree/110d037f388ad596916310179441b60453c042fd