**Student name:** Sakshi Maurya

**Student Id:** 11801511

**Roll no:**. RK18CJA17

**Question number assigned**: Q17

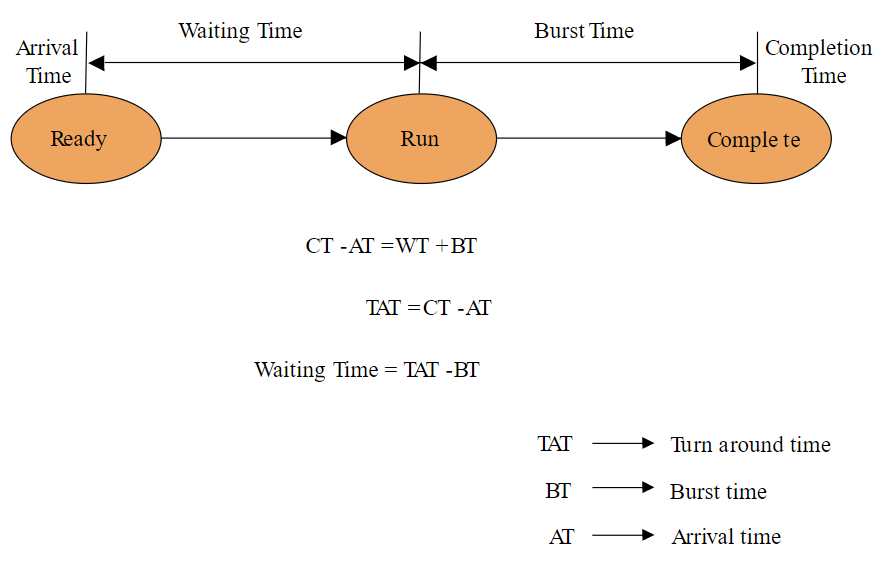
**Email address:** [sakshi3092maurya@gmail.com](mailto:sakshi3092maurya@gmail.com)

**Github Link:** <https://github.com/Sakshi1329/Os-Simulation>

1. **Explain the problem in terms of operating system concept?**

In this problem, one has to create a program to compute the average waiting time i.e. the amount of time a process has been waiting in the ready queue and average turnaround time i.e. the amount of time elapsed from the time of submission to the time of completion, where you have to design a scheduler following non-preemptive scheduling approach which means , once the resources i.e. CPU cycles is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state. It does not interrupt a process running CPU in middle of the execution. Instead , it waits till the process complete its CPU burst time and then it can allocate the CPU to another process. In this task, it schedules the processes that arrives at different units and having burst time i.e the amount of time required by a process for executing on CPU double the arrival time i.e the time at which the process arrives in the ready queue.Here the scheduler selects the process with largest burst time form the queue for the execution.

In addition to the task assigned, also compute the average waiting time if the processes are executed according to Shortest Job First scheduling approach with the same attribute values.in this approach , the process having smallest execution time is chosen for the next execution.



**Average waiting time** =

total\_waiting\_time / no\_of\_processes

**turn around time**

= waiting\_time + burst\_time for all processes

**Average turn around time** =

total\_turn\_around\_time / no\_of\_processes

1. **Write the algorithm for proposed solution of the assigned problem?**

Algorithm used only to sort the elements with high burst time i.e. Selection sort (descending order). And for the shortest job first one has again used to sort the elements using Selection sort (ascending order).The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two sub arrays in a given array:

1) The sub array which is already sorted.  
2) Remaining sub array which is unsorted.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted sub array is picked and moved to the sorted sub array.

1. **Calculate complexity of implemented algorithm.**

Algorithm sort the array in ascending order.

SelectionSort(array, size)

repeat (size - 1) times

set the first unsorted element as the minimum

for each of the unsorted elements

if element < currentMinimum

set element as new minimum

swap minimum with first unsorted position end selectionSort

**COMPLEXITY**

| Cycle | Number of Comparison |
| --- | --- |
| 1st | (n-1) |
| 2nd | (n-2) |
| 3rd | (n-3) |
| ... | ... |
| last | 1 |

****Number of comparisons:****(n-1) + (n-2) + (n-3) +.....+ 1 = n(n-1)/2 nearly equals to n2  
****Complexity**** = O(n2)

Also, we can analyze the complexity by simply observing the number of loops. There are 2 loops so the complexity is n\*n=n2

Similarly , for the descending order . The complexity is o(n2).

1. **Explain all the constraints given in the problem. Code snippet**

*//Header files*

#include<stdio.h>

#include<conio.h>

*//main function*

int main()

{

/*/variables declaration*

int bt[20],at[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg\_wt,avg\_tat;

float avg\_wtt, avg\_tatt;

*//total number of process*

printf("Enter Total Number of Process:");

scanf("%d",&n);

*//entering arrival time and calculation of burst time*

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

{

printf("\n P[%d]\n",i+1);

printf("Arrival time:");

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

bt[i]=2\*at[i];

printf("Burst time:%d\n",bt[i]);

p[i]=i+1;

}

*//sorting the burst time*

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

{

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

{

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

{

temp=at[j];

at[j]=at[i];

at[i]=temp;

temp=bt[j];

bt[j]=bt[i];

bt[i]=temp;

temp=p[j];

p[j]=p[i];

p[i]=temp;

}

}

}

*//inital process waiting time is zero*

wt[0]=0;

*//calculation of waiting time for rest of the processes*

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

{

wt[i]=0;

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

wt[i]+=bt[j];

total+=wt[i];

}

*//calculating average waiting time for the processes*

avg\_wt=total/n;

total=0;

*//to display the output in tabular form*

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\t\t\t\t\t\t\t RESULT:-");

printf("\n\t\t\t\t\t Non-Preemptive\n\n");

printf("\n\t\t------------------------------------------------------------------------------------");

printf("\n\t\t|PROCESS| \tAT\t| \tBT\t| \tWT\t| \tTAT\t|" );

printf("\n\t\t------------------------------------------------------------------------------------");

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

{

*//calcuation of turnaround time of process*

tat[i]=bt[i]+wt[i];

total+=tat[i];

printf("\n\t\t|p[%d]\t|\t%d\t|\t%d\t|\t%d\t|\t%d\t| ",p[i],at[i],bt[i],wt[i],tat[i]);

*//for output in tabular form*

}

printf("\n\t\t------------------------------------------------------------------------------------");

*//calcution of average turnaround time for the processes*

avg\_tat=total/n;

*//Gantt chart*

printf("\n\t\t\t\t ..Order of execution according to Gantt chart..\n\n");

printf("\t\t\t\t\t--------------\n\t\t\t\t\t");

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

{

printf("| Process[%d] |\n\t\t\t\t\t",p[i]);

}

*//printing the output of average waiting and turnaround time*

printf("\n\nAverage Waiting Time=%d",avg\_wt);

printf("\nAverage Turnaround Time=%d\n\n\n\n\n",avg\_tat);

*//using short job first scheduling*

*//sorting the burst time*

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;

temp=at[i];

at[i]=at[pos];

at[pos]=temp;

}

*//inital value of waiting time of first process*

wt[0]=0;

*//calculation of waiting time of rest of the processes*

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

{

wt[i]=0;

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

wt[i]+=bt[j];

total+=wt[i];

}

*//calculation of average waiting time for the processes*

avg\_wtt=(float)total/n;

total=0;

*//tabular form of the result*

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\t\t\t\t\t\t\t RESULT:-");

printf("\n\t\t\t\t\t Short Job First, Non-Preemptive\n\n");

printf("\n\t\t------------------------------------------------------------------------------------");

printf("\n\t\t|PROCESS| \tAT\t| \tBT\t| \tWT\t| \tTAT\t|" );

printf("\n\t\t------------------------------------------------------------------------------------");

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

{

*//calcuation of turnaround time of process*

tat[i]=bt[i]+wt[i];

total+=tat[i];

printf("\n\t\t|p[%d]\t|\t%d\t|\t%d\t|\t%d\t|\t%d\t| ",p[i],at[i],bt[i],wt[i],tat[i]); *//output form*

}

printf("\n\t\t------------------------------------------------------------------------------------");

*//calcution of average turnaround time for the processes*

avg\_tatt=(float)total/n;

*//Gantt cart*

printf("\n\t\t\t\t ..Order of execution according to Gantt chart..\n");

printf("\t\t\t\t\t--------------\n\t\t\t\t\t");

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

{

printf("| Process[%d] |\n\t\t\t\t\t",p[i]);

}

*//Output for average waiting and turnaround time*

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

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

return 0;

}

In this problem, we implemented simply allow the user to input the umber of the processes and arrival time. In the first half of the code, it sorts according to the larger burst time then calculate the average waiting time and average turnaround time. Then, on the second half of the code where we used the shortest job first scheduling (SJFS)we have used to sort the elements with minimum burst time i.e. in ascending order. Then ,calculated the average waiting time and average turnaround time. Here , in SJFS, it improves process throughput by making sure that shorter jobs are executed first, hence possibly have a short turnaround time.

1. **If you have implemented any additional algorithm to support the solution, explain the need and usage of the same.**

Selection sort-In Selection sort, the smallest element is exchanged with the first element of the unsorted list of elements (the exchanged element takes the place where smallest element is initially placed). Then the second smallest element is exchanged with the second element of the unsorted list of elements and so on until all the elements are sorted.

It is basically used to  find the minimum element in the array and swap it with the element in the 1st position. Find the minimum element again in the remaining array[2, n] and swap it with the element at 2nd position, now we have two elements at their correct positions. We have to do this n-1 times to sort the array.

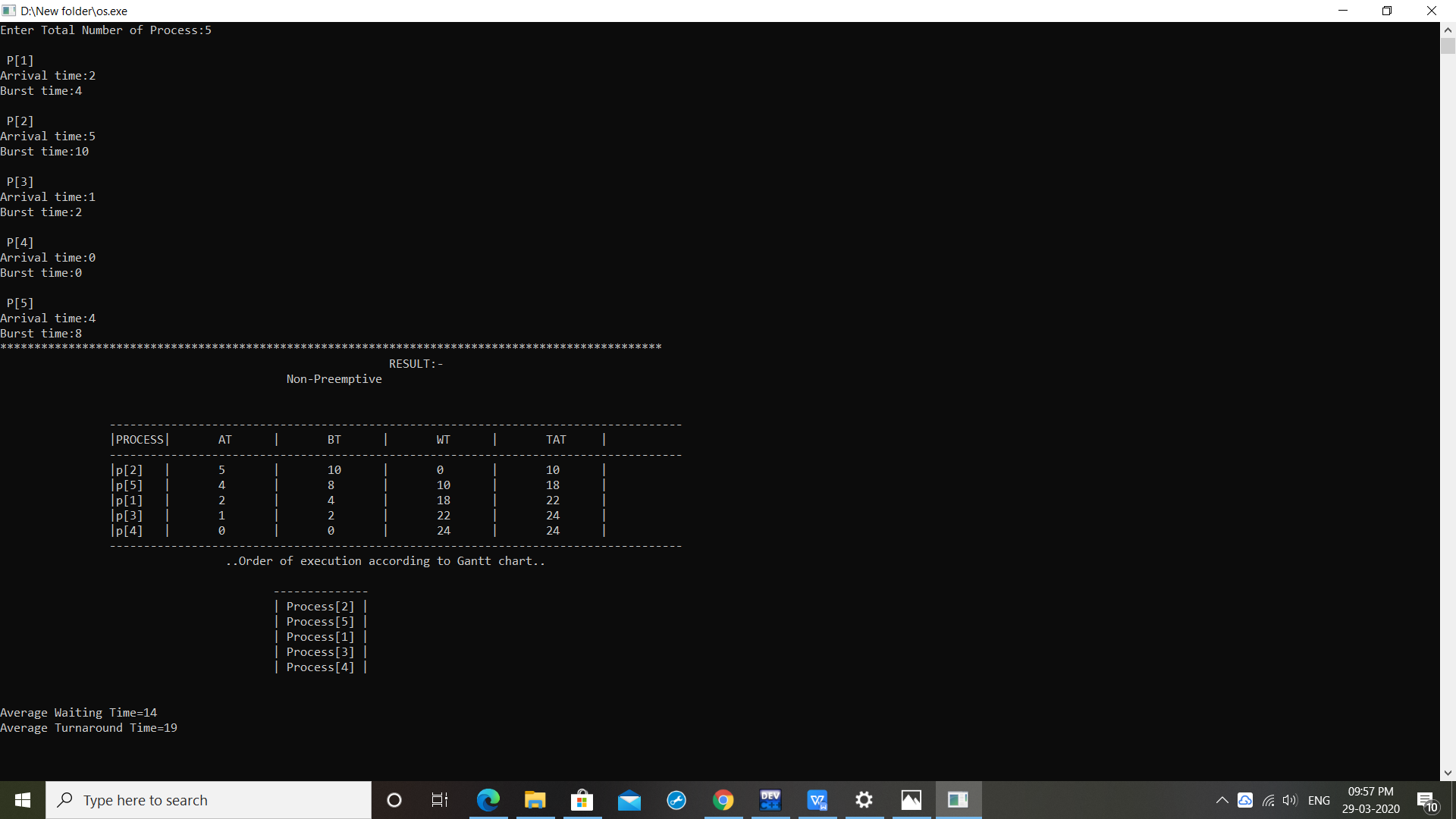
1. **Explain the boundary conditions of the implemented code**

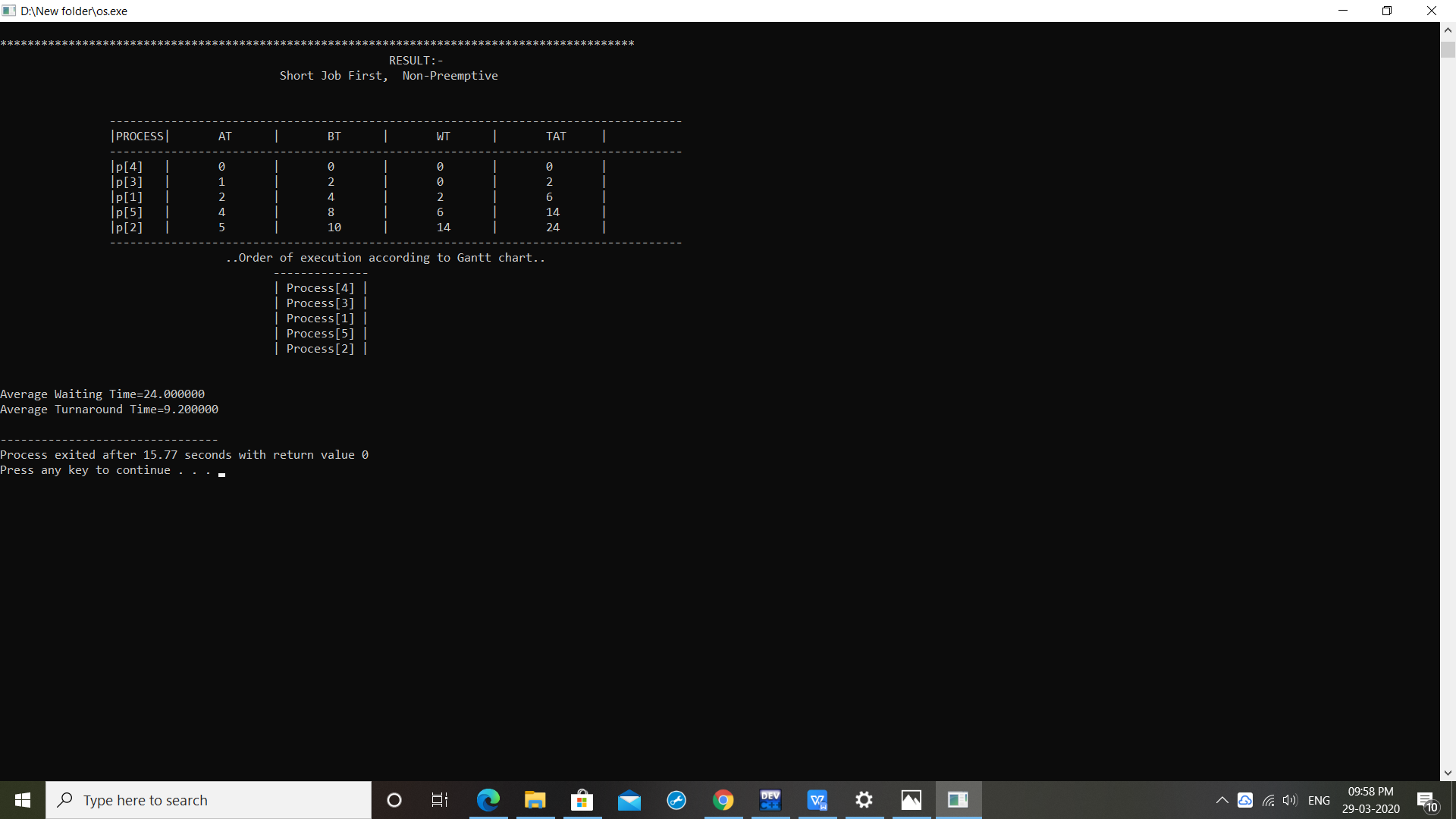
* As the program used be non preemptive scheduling approach where the once a process enters the running state, it cannot be preempted until it completes its allotted time.
* The burst time should be twice the arrival time that is, the input entered by the user for the arrival time. The code has to calculated for m the inputted arrival time, to find the value of burst time.
* Scheduler selects the process with largest burst time for m the queue for the execution. But it won’t be used in the Shortest job first scheduling as, it selects the process with minimum burst time.

1. **Explain all the test case applied on the solution of assigned problem**

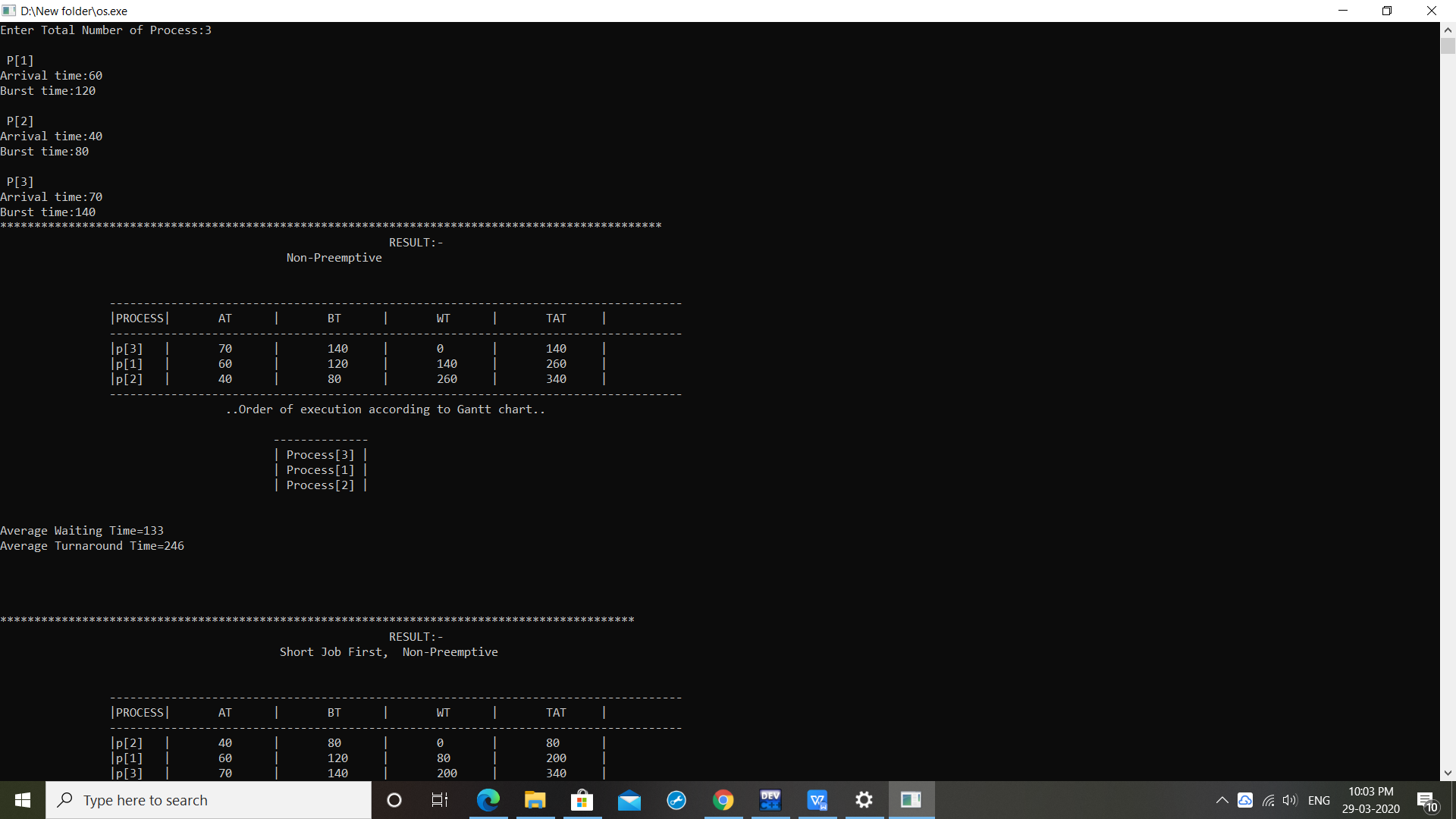
TEST CASE-1

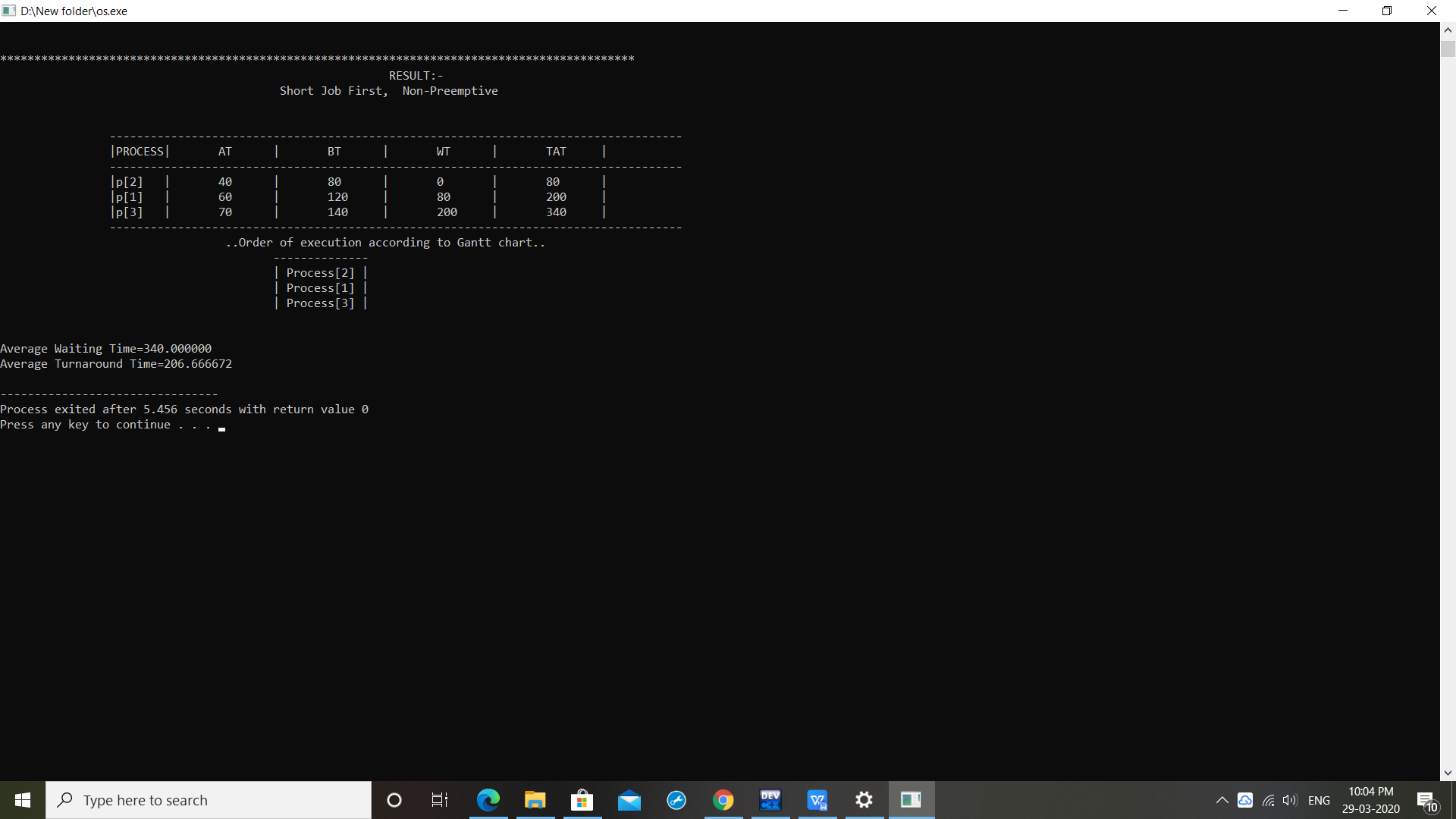
5 processes with arrival time: 2,5,1,0,4 respectively. The following is the output:





TEST CASE -2

3 processes with arrival time 60,40,70 respectively. The following is the output: 



1. **Have you made minimum 5 revisions of solution on Github?**

Github link: <https://github.com/Sakshi1329/Os-Simulation>