Course Title:- Operating Systems

Course code:-CSE-316

Name:- Devesh Kumawat

Registration number:- 11713770/B-54

Section:-EE028

Faculty:-Manjit Kaur

E-Mail Address:-deveshkumawat167@gmail.com

GitHubLink:- https://devesh3770.github.io/Devesh-OS-assignment/

Question no 1 : -

Develop a scheduler which submits the processes to the processor in the following scenario,and compute the scheduler performance by providing the waiting time for process,turn around time for process and average waiting time and turn around time. Considering the arrival time and the burst time requirement of the processes the scheduler schedules the processes by interrupting the processor after every 3 units of time and does consider the completion of the process in this iteration. The schedulers than checks for the number of processes waiting for the processor and allots the processor to the process but interrupting the processor after every 6 units of time and considers the completion of the process in this iteration. The scheduler after the second iteration checks for the number of processes waiting for the processor and now provides the processor to the process with the least time requirement to go in the terminated state.

The inputs for the number of requirements, arrival time and burst time should be provided by the user.

Consider the following units for reference.

Process	Arrival time	Burst time
P1	**0**	**18**
P2	**2**	**23**
P3	**4**	**13**
P4	**13**	**10**

Description:-

CPU/Processor Scheduling:

1. It is basic activity for multiprogrammed Operating system
2. Selecting an appropriate distribution of CPU and I/O bound programs is crucial for CPU scheduling
3. CPU scheduling is done by the Short Term Scheduler/ CPU Scheduler

4. CPU scheduler decides which of the processes in the ready queue is to be allocated the CPU.
5. Ready queue can be maintained either as a FIFO queue, Priority queue, tree of any other unordered linked list.
6. CPU scheduling can be preemptive or Non preemptive type.

Non-Preemptive Scheduling:

Here the process is allocated the CPU, It keeps the processor with it till it release the processor voluntarily either by terminating or by switching to waiting state.

Preemptive Scheduling:

Here a process allocated the CPU may be required to release the control on occurrence of interrupt or another process completes(considering priority, time slice etc)

Dispatcher:

Another component that is involved in the CPU-scheduling function is the dispatcher, which is the module that gives control of the CPU to the process selected by the short-term scheduler. It receives control in kernel mode as the result of an interrupt or system call. Context switches, in which the dispatcher saves the state (also known as context) of the process or thread that was previously running; the dispatcher then loads the initial or previously saved state of the new process.

Switching to user mode.

Jumping to the proper location in the user program to restart it.

Scheduling Criteria:

They are parameters or metrics or characteristics used for comparing different scheduling.
Different CPU scheduling criteria are include

CPU Utilization: It measures the CPU usages in terms of how busy the processor is or load on the processor. It is with respect to the system.

Throughput: It measures the work being done. It is the numbers of processes that are completed per time unit. It is with respect to the system.

Turnaround Time(TAT):

1. It measures the time taken to execute a process
2. It is the interval of time between the submission time of the process to its completion time.
3. It is the difference in the completion and submission times of a process.
4. It is limited by the speed of the output devices.
5. It is with respect to a process.

Formula for TAT:

$TAT_i = CT_i - AT_i$

Where CT_i=Completion Time

AT_i= Arrival Time

Waiting Time(WT):

1. It measures the times a process waits in the ready queue.
2. It is the sum of amount of time spent waiting in the ready queue by the process.
3. It is directly affected by the type of CPU scheduling algorithm
4. It is with respect to a process.

Formula for WT:

Let TAT_i be the turnaround time and BT_i be the CPU burst time respectively of a process Pi. Then the waiting time of the process is:

$WT_i = TAT_i - BT_i$

Response Time(RT):

1. It is a measure of time taken to produce the first response for a process.

2. It is the time between submission of a request and the generation of first response.
3. It is a metric for interactive/time sharing systems.
4. It is required to reduce variance in these times for these systems.
5. In summary, it is desirable to maximize CPU utilization and throughput and minimize turnaround time, waiting time and response time.
6. Average time for turnaround, waiting can also be used.

Gantt Chart: Gantt Chart

- It is a rectangular time scale diagram with x-axis depicting the time line.
- It is used to represent the scheduling of processes graphically.

SCHEDULING ALGORITHMS

CPU scheduling deals with the problem of deciding which of the processes in the ready queue is to be allocated the CPU. CPU Scheduling algorithms are :

Non-preemptive Algorithm:

- First come First served (FCFS)
- Priority(Non-preemptive)
- Shortest Job First(SJF)

Preemptive Algorithm:

- Shortest remaining Time First(SRTF)
- Round Robin(RR)

Round-Robin Scheduling: -

Round-robin (RR) is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices (also known as time quanta) are assigned to each process in equal portions and in circular order handling all processes without priority starvation-free. Round-robin scheduling can be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an operating system concept.

Algorithm:-

Steps to find waiting times of all processes:

1- Create an array rem_bt[] to keep track of remaining

burst time of processes. This array is initially a

copy of bt[] (burst times array)

2- Create another array wt[] to store waiting times

of processes. Initialize this array as 0.

3- Initialize time : t = 0

4- Keep traversing the all processes while all processes

are not done. Do following for i'th process if it is

not done yet.

a- If rem_bt[i] > quantum

(i) t = t + quantum

(ii) bt_rem[i] -= quantum;

c- Else // Last cycle for this process

(i) t = t + bt_rem[i];

(ii) wt[i] = t - bt[i]

(ii) bt_rem[i] = 0; // This process is over

Code :-

```
#include<iostream>

using namespace std;

void findWaitingTime(int processes[], int n, int bt[],

 int wt[], int at[])

{

 int service_time[n];

 service_time[0] = 0;

 wt[0] = 0;

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

 {

 service_time[i] = service_time[i-1] + bt[i-1];

 wt[i] = service_time[i] - at[i];

  if (wt[i] < 0)

 wt[i] = 0;

 }

}

void findTurnAroundTime(int processes[], int n, int bt[],

 int wt[], int tat[])
```

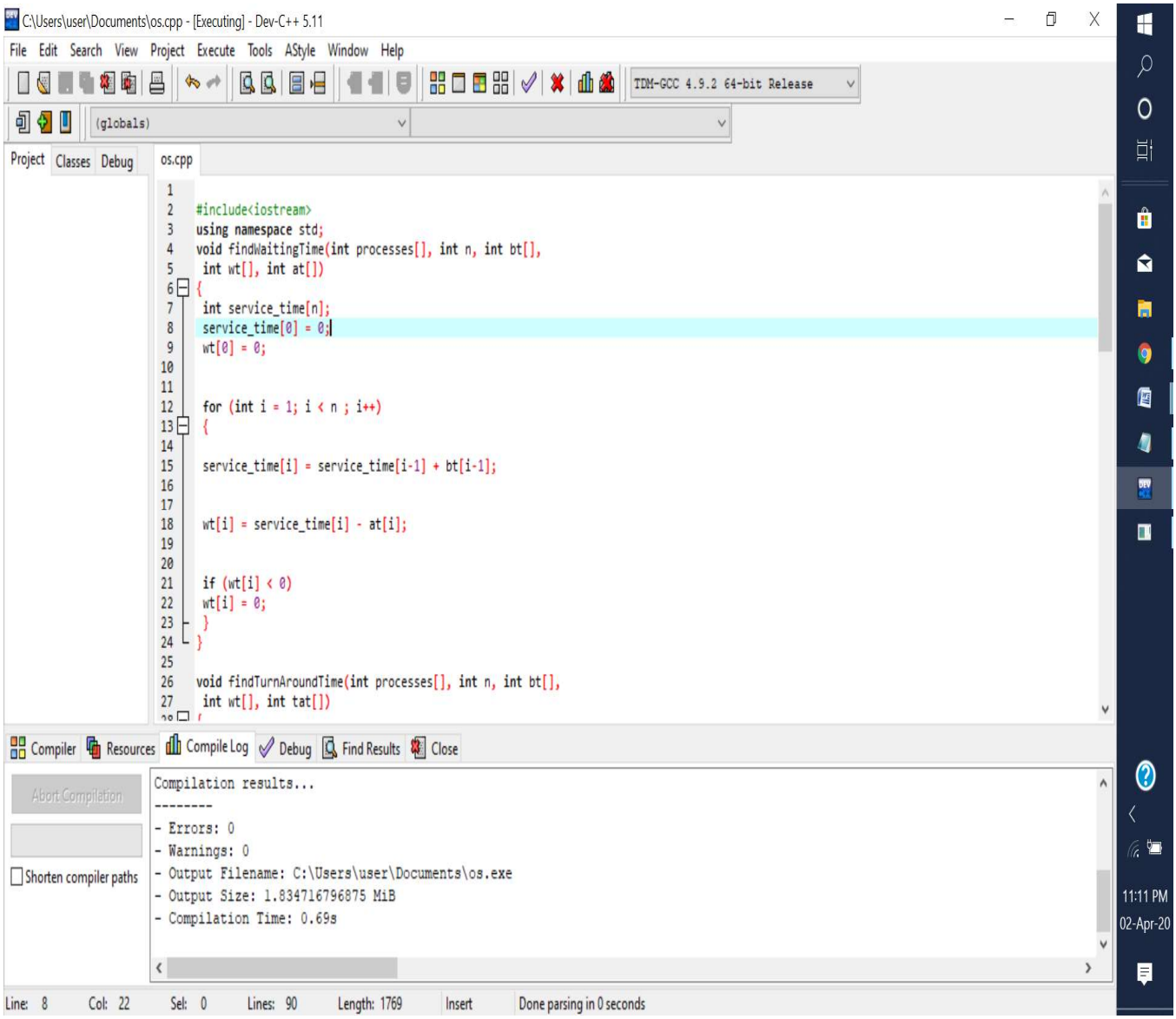
```
{
 for (int i = 0; i < n ; i++)
 tat[i] = bt[i] + wt[i];
}
void findavgTime(int processes[], int n, int bt[], int at[])
{
 int wt[n], tat[n];
findWaitingTime(processes, n, bt, wt, at);
findTurnAroundTime(processes, n, bt, wt, tat);
cout << "Processes " << " Burst Time " << " Arrival Time "
 << " Waiting Time " << " Turn-Around Time "
 << " Completion Time \n";
 int total_wt = 0, total_tat = 0;
 for (int i = 0 ; i < n ; i++)
 {
 total_wt = total_wt + wt[i];
 total_tat = total_tat + tat[i];
 int compl_time = tat[i] + at[i];
 cout << " " << i+1 << "\t\t" << bt[i] << "\t\t"
```

```
<< at[i] << "\t\t" << wt[i] << "\t\t "
<< tat[i] << "\t\t " << compl_time << endl;
}
cout << "Average waiting time = "

<< (float)total_wt / (float)n;

cout << "\nAverage turn around time = "

<< (float)total_tat / (float)n;
}
int main()
{
 int n,i;
 cout<<"Enter the no.of Processes"<<endl;
 cin>>n;

 int processes[n],burst_time[n],arrival_time[n];
```

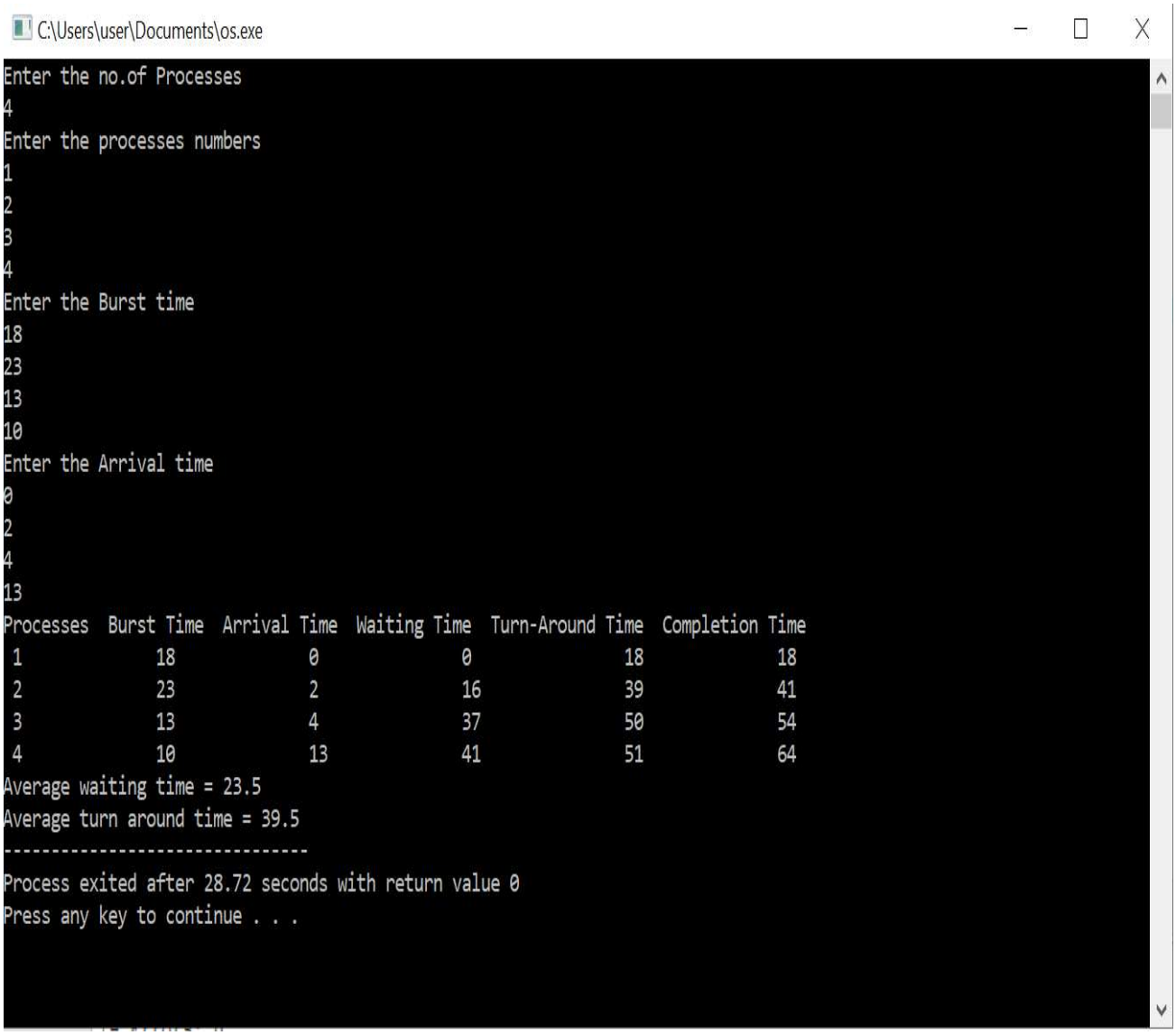
```
 cout<<"Enter the processes numbers"<<endl;
 for(i=0;i<n;i++){

cin>>processes[i];
 }
 cout<<"Enter the Burst time"<<endl;
 for(i=0;i<n;i++){
 cin>>burst_time[i];
 }
 cout<<"Enter the Arrival time"<<endl;
 for(i=0;i<n;i++){

 cin>>arrival_time[i];
 }
 findavgTime(processes, n, burst_time, arrival_time);
 return 0;
}
```

Complexity:- O (n) complexity.

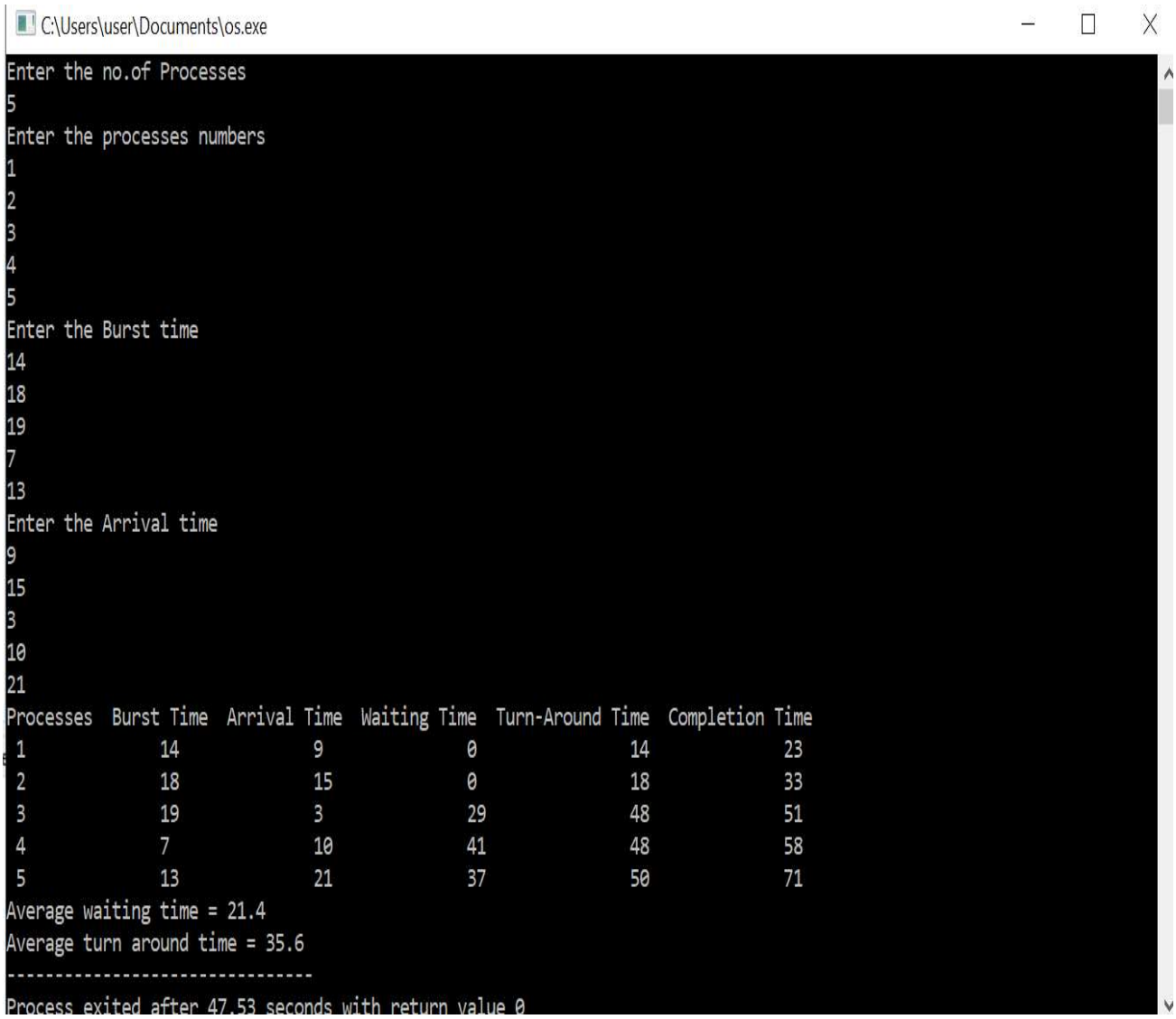
Compilation Results:-



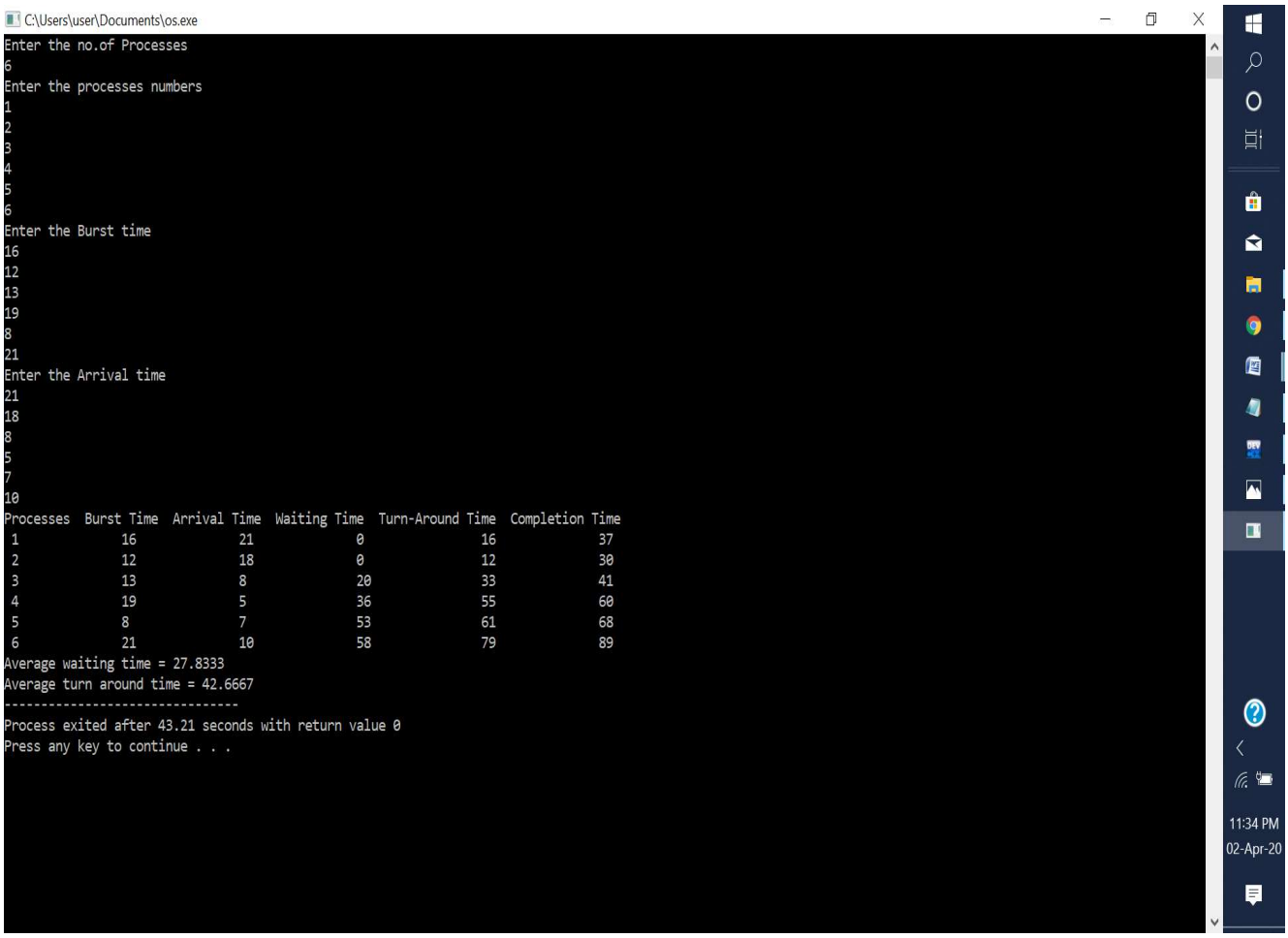
Code Output 1:-



Code Output 2:-



Code Output 3:-



GitHub Link:- https://devesh3770.github.io/Devesh-OS-assignment/

References:-

- https://www.cs.yale.edu/homes/aspnes/pinewiki/ProcessorScheduling.html
- http://codequiz.in/cpu-scheduling/
- https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/