OS Project

Student Name : Sujeet Kumar Jaiswal			
Student ID	: 11712230	**Roll No**	:29
Email Address: sujeetsahilbxr@gmail.com			
GitHub Link	**: https://github.com/sujeetsahiljais/OS_Project**		

===

Question Description

Question :

Considering 4 processes with 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 then 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

Develop a scheduler which submits the processes to the processor in the above defined scenario, and compute the scheduler performance by providing the waiting time for process, turnaround time for process and average waiting time and turnaround time.

===

Algorithm:

```
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]
                       •    bt_rem[i] = 0; // This process is over
```

===

- **Description (Purpose of Use):**

  - CPU scheduling algorithm is a process where each process is assigned a fixed time slot in

  - cyclic way.

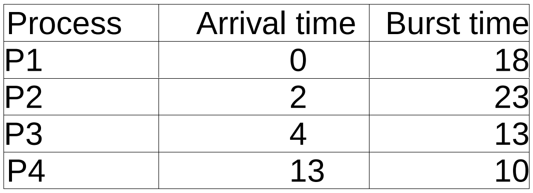
  - It is simple, easy to implement, and starvation-free as all processes get fair share of CPU.

  - One of the most commonly used technique in CPU scheduling as a core.

  - It is preemptive as processes are assigned CPU only for a fixed slice of time at most.

  - The disadvantage of it is more overhead of context switching.

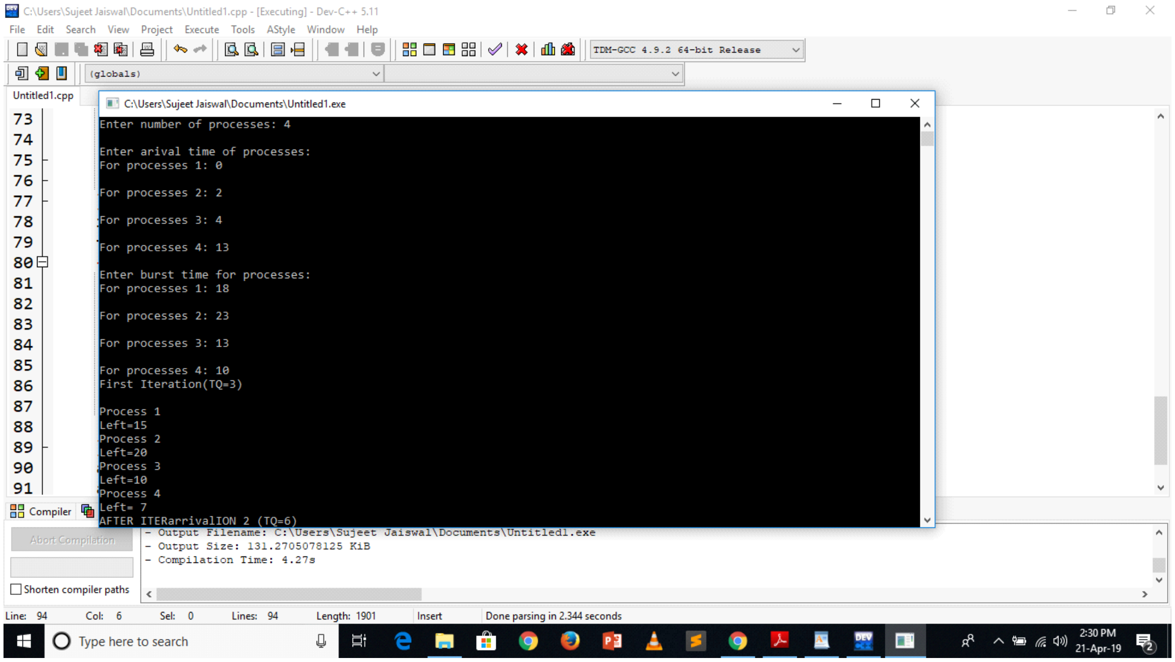
Given Test Cases:



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

Code snippet:

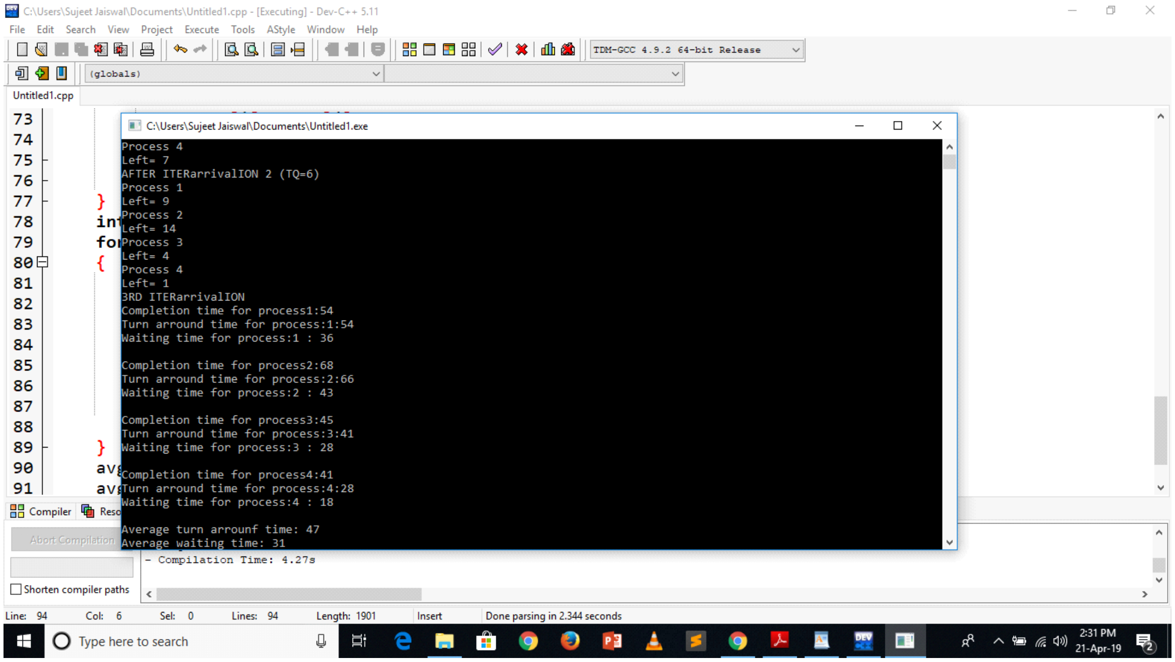
```
#include<stdio.h>
int main()
{
        int i,m;
        printf("Enter number of processes: ");
        scanf("%d",&m);
        int arrival[m];
        int burst[m],burst1[m];
        int waiting[m];
        int tarrival[m];
        int tq1=3;
        int tq2=6;
        int avgT=0;
        int avgW=0;
        printf("\nEnter arival time of processes: ");
        for(i=0;i<m;i++)
        {       printf("\nFor processes %d: ",i+1);
                scanf("%d",&arrival[i]);
        }
        printf("\nEnter burst time for processes: ");
        for(i=0;i<m;i++)
        {
                printf("\nFor processes %d: ",i+1);
                scanf("%d",&burst[i]);
                burst1[i]=burst[i];
        }
        printf("First Iteration(TQ=3)\n");
        for(i=0;i<m;i++)
        {
                if(arrival[i]<=tq1+arrival[i-1])
                {
                        burst[i]=burst[i]-tq1;
                        printf("\nProcess %d ",i+1);
                        printf("\nLeft=%d ",burst[i]);
                }
                else
                {
                burst[i]=burst[i]-tq1;
                printf("\nProcess %d ",i+1);
                printf("\nLeft= %d",burst[i]);
                }
        }
        printf("\nAFTER ITERarrivalION 2 (TQ=6) ");
        for(i=0;i<m;i++)
        {
                if(arrival[i]<=tq2+arrival[i-1])
                {
                        burst[i]=burst[i]-tq2;
                        printf("\nProcess %d ",i+1);
                        printf("\nLeft= %d ",burst[i]);
                }
                else
                {
                        burst[i]=burst[i]-tq2;
                        printf("\nProcess %d ",i+1);
                        printf("\nLeft= %d ",burst[i]);
                }
        }
        printf("\n3RD ITERarrivalION");
        int j,temp;
        for(i=0;i<m;i++)
        {
                for(j=i+1;j<m;j++)
                {
                        if(burst[i]>burst[j])
                        {
                                temp=burst[i];
                                burst[i]=burst[j];
                                burst[j]=temp;
                        }
                }
        }
        int ct[4]={54,68,45,41};
        for(i=0;i<m;i++)
        {
                tarrival[i]=ct[i]-arrival[i];
                waiting[i]=tarrival[i]-burst1[i];
                printf("\nCompletion time for process%d:%d \n",i+1,ct[i]);
                printf("Turn arround time for process:%d:%d \n",i+1,tarrival[i]);
                printf("Waiting time for process:%d : %d \n",i+1,waiting[i]);
                avgT=avgT+tarrival[i];
                avgW=avgW+waiting[i];
        }
        avgT=avgT/m;
        avgW=avgW/m;
        printf("\nAverage turn arrounf time: %d",avgT);
        printf("\nAverage waiting time: %d",avgW);
}
```


C:\Users\Sujeet Jaiswal\Documents\Untitled1.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Untitled1.cpp
C:\Users\Sujeet Jaiswal\Documents\Untitled1.exe
Compiler
Abort Compilation
Shorten compiler paths
Type here to search
2:30 PM
21-Apr-19

```
Enter number of processes: 4

Enter arival time of processes:
For processes 1: 0

For processes 2: 2

For processes 3: 4

For processes 4: 13

Enter burst time for processes:
For processes 1: 18

For processes 2: 23

For processes 3: 13

For processes 4: 10
First Iteration(TQ=3)

Process 1
Left=15
Process 2
Left=20
Process 3
Left=10
Process 4
Left= 7
AFTER ITERarrivalION 2 (TQ=6)
```

```
- Output Filename: C:\Users\Sujeet Jaiswal\Documents\Untitled1.exe
- Output Size: 131.2705078125 KiB
- Compilation Time: 4.27s
```

```
Line: 94   Col: 6   Sel: 0   Lines: 94   Length: 1901   Insert   Done parsing in 2.344 seconds
```


C:\Users\Sujeet Jaiswal\Documents\Untitled1.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Untitled1.cpp
C:\Users\Sujeet Jaiswal\Documents\Untitled1.exe
Compiler
Abort Compilation
Shorten compiler paths
Type here to search
2:31 PM
21-Apr-19

```
Process 4
Left= 7
AFTER ITERarrivalION 2 (TQ=6)
Process 1
Left= 9
Process 2
Left= 14
Process 3
Left= 4
Process 4
Left= 1
3RD ITERarrivalION
Completion time for process1:54
Turn arround time for process:1:54
Waiting time for process:1 : 36

Completion time for process2:68
Turn arround time for process:2:66
Waiting time for process:2 : 43

Completion time for process3:45
Turn arround time for process:3:41
Waiting time for process:3 : 28

Completion time for process4:41
Turn arround time for process:4:28
Waiting time for process:4 : 18

Average turn arrounf time: 47
Average waiting time: 31
```

```
- Compilation Time: 4.27s
```

```
Line: 94   Col: 6   Sel: 0   Lines: 94   Length: 1901   Insert   Done parsing in 2.344 seconds
```

Refer this link

https://github.com/sujeetsahiljais/OS_Project

Thank You