Student Name: Akash Katiyar
Student ID: 11801431 (A26)
Email Address: Akashkatiyar08@gmail.com
GitHub Link: https://github.com/Kfire08/OS_Schedular_assignment

Problem: 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.

Answer: We can solve this question by using two algorithms of Operating System:

1. **Round Robin Scheduling Algorithm**: for fixed time slice i.e. for 3 unit and 6 unit of time.
2. **Shortest Job First Scheduling Algorithm**: for providing the scheduler to the process with least execution time.

Algorithm:

Step 1: Start the process
Step 2: Accept the number of processes in the ready Queue along with arrival time and burst time.
Step 3: For each process in the ready Q, assign the process with a time slice of 3 unit in first iteration.
Step 4: For each process again assign the process with a time slice of 6 unit in second iteration.
Step 5: Sort the left burst time in ascending order and assign Processes with least execution time i.e. CPU is assigned to process with less CPU burst time.
Step 6: Calculate

(a) Completion time(1) = 9*no of process + process number + left burst time(1).
(b) Completion time(n) = completion time(n-1) + left burst time(n).
(c) Turnaround time for process(n) = Completion time(n) - arrival time(n).
(d) Waiting time for process(n) =turnaround time(n) – total burst time(n).
Step 7: Calculate
(a) Average waiting time = Total waiting Time / Number of process
(b) Average Turnaround time = Total Turnaround Time / Number of process
Step 8: Stop the process

Description:

To implement the above problem we have to make three iterations in which first iteration with a time slice of 3 units reduce the burst time of each process by 3. In second iteration with a time slice of 6 units reduce the burst time of each process by 6. In third iteration sort the left burst time in ascending order and calculate turnaround time and waiting time for the available process sequence and then calculate average waiting time and turnaround time using above formulas.

Code:

```
//Question no -1 Akash Katiyar(A26)
#include<stdio.h>
int main()

{
int allocation,cpu,pno[10],completion[10];
int
i,j,a,p=0,count,no_of_process,time,remaining,flag=0,time_quantum1=3,time_quantum
2=6;

int
waiting_time=0,turnaround_time=0,arrival_time[10],burst_time[10],new_burst_time[1
0],new_burst_time2[10],bt[10],response_time[10];

printf("Enter Total Process:\t ");

scanf("%d",&no_of_process);

remaining=no_of_process;

for(count=0;count<no_of_process;count++)

{
printf("Enter Arrival Time and Burst Time for Process Number %d :",count+1);
scanf("%d",&arrival_time[count]);
scanf("%d",&burst_time[count]);
bt[count]=burst_time[count];

}
        printf("Burst times after first iteration\n");
        for(count=0;count<no_of_process;count++)

{
        p=p+1;
        printf("P[%d] - %d\n",p,burst_time[count]-time_quantum1);
```

```
        burst_time[count]=burst_time[count]-time_quantum1;
}
   p=0;
        printf("\nBurst times after second iteration\n");
for(count=0;count<no_of_process;count++)

{
   p=p+1;
   printf("P[%d] - %d\n",p,burst_time[count]-time_quantum2);
   new_burst_time[count]=burst_time[count]-time_quantum2;
   new_burst_time2[count]=burst_time[count]-time_quantum2;
}
 for (i = 0; i < no_of_process; ++i)
        {
           for (j = i + 1; j < no_of_process; ++j)
           {
              if (new_burst_time[i] > new_burst_time[j])
              {
                 a =  new_burst_time[i];
                 new_burst_time[i] = new_burst_time[j];
                                    new_burst_time[j] = a;
              }
           }

        }


        for (i = 0; i < no_of_process; ++i)
        {
           for (j = i + 1; j < no_of_process; ++j)
           {
              if (burst_time[i] > burst_time[j])
              {
                 a =  burst_time[i];
                 burst_time[i] = burst_time[j];
                                    burst_time[j] = a;
              }
           }
        }

       for (i = 0; i < no_of_process; ++i)
        {
           for (j = 0; j < no_of_process+1; ++j)
           {
              if (new_burst_time[i] == new_burst_time2[j])
              {
                allocation=j;
              }
           }
           pno[i]=allocation;
        }
       cpu=(9*no_of_process)+pno[0]+1;
       p=0;
       completion[0]=cpu+new_burst_time[0];
```

```
        for(i=0;i<no_of_process;i++)
        {
          completion[i+1]=completion[i]+new_burst_time[i+1];
            }
    printf("\nThird iteration\n");
    printf("\n\nAllocation Sequence|Turnaround Time|Waiting Time\n");
    printf("-------------------+--------------+-----------\n");
    for (i=0;i<no_of_process;i++)
    {

    printf("P[%d]\t\t  |\t%d\t  |\t%d\n",pno[i]+1,completion[i]-
    arrival_time[pno[i]],completion[i]-arrival_time[pno[i]]-bt[pno[i]]);

    waiting_time+=completion[i]-arrival_time[pno[i]]-bt[pno[i]];
    turnaround_time+=completion[i]-arrival_time[pno[i]];
    }
    printf("\nAverage Waiting Time= %f\n",waiting_time*1.0/no_of_process);

    printf("Avg Turnaround Time = %f",turnaround_time*1.0/no_of_process);

    return 0;
    }
}
```

Output:

```
C:\Users\Akash\Desktop\OS_Schedular.exe
Enter Total Process:     4
Enter Arrival Time and Burst Time for Process Number 1 :0 18
Enter Arrival Time and Burst Time for Process Number 2 :2 23
Enter Arrival Time and Burst Time for Process Number 3 :4 13
Enter Arrival Time and Burst Time for Process Number 4 :13 10
Burst times after first iteration
P[1] - 15
P[2] - 20
P[3] - 10
P[4] - 7

Burst times after second iteration
P[1] - 9
P[2] - 14
P[3] - 4
P[4] - 1

Third iteration


Allocation Sequence|Turnaround Time|Waiting Time
-------------------+--------------+-----------
P[4]               |    28        |    18
P[3]               |    41        |    28
P[1]               |    54        |    36
P[2]               |    66        |    43

Average Waiting Time= 31.250000
Avg Turnaround Time = 47.250000
--------------------------------
```