Experiment – 7

NAME : **Madhuram Brijeshkumar Modi**

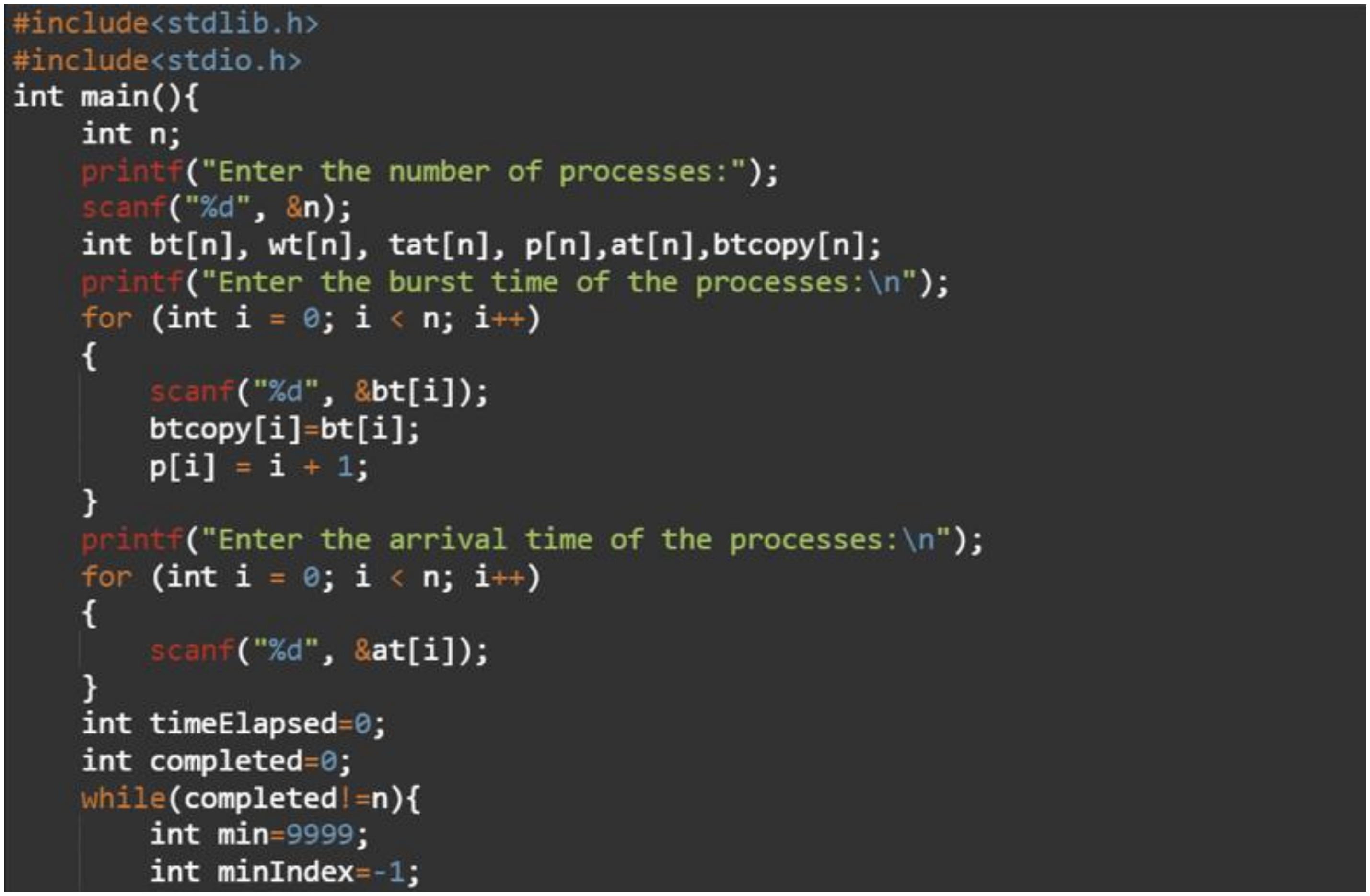
ROLL NO. : **21BCP102**

DIV, GROUP : **2,G3**

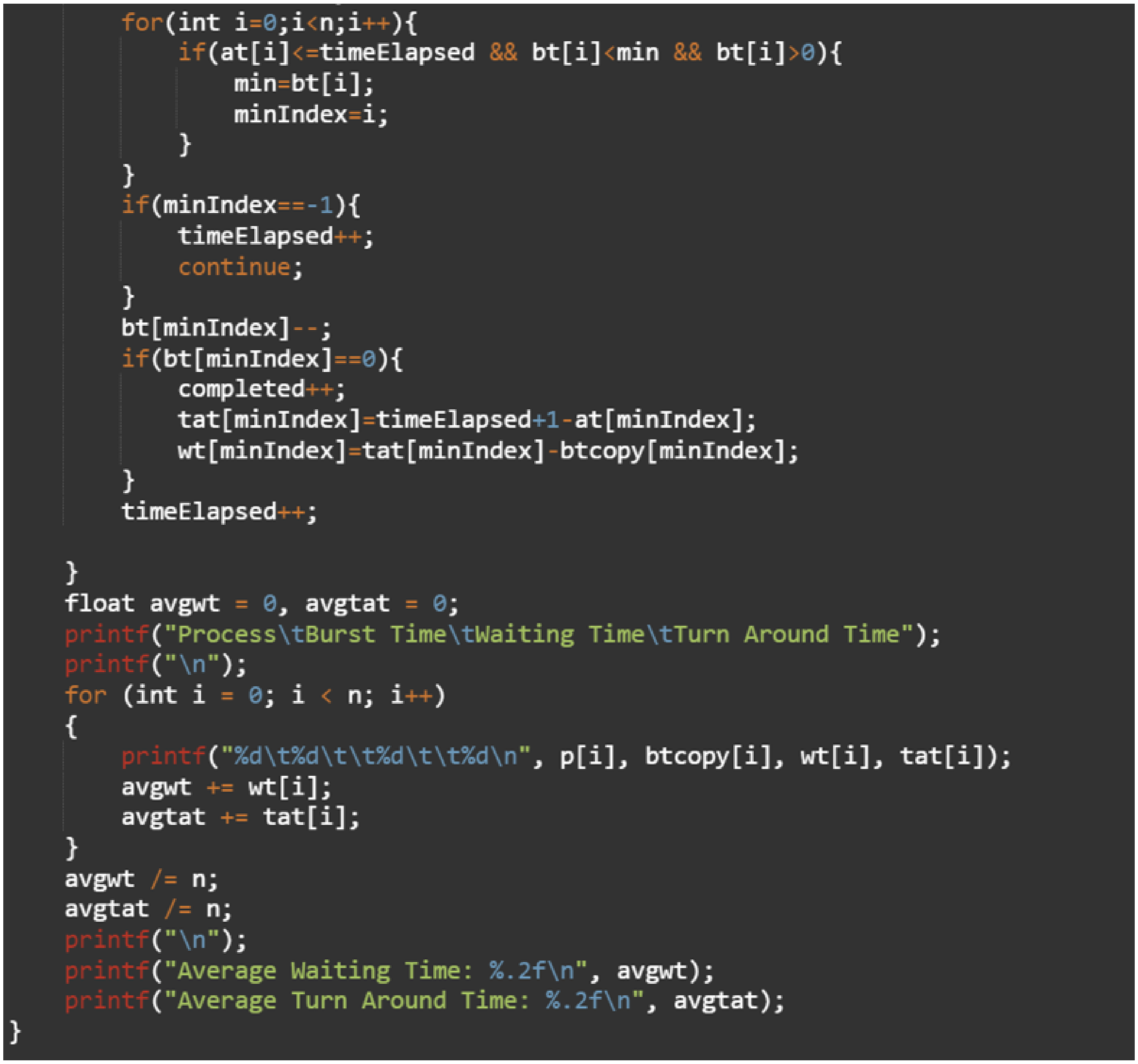
Scheduling Algorithms -2

3. Shortest Remaining Time Next

It is the preemptive form of SJF. In this algorithm, the OS schedules the Job according to the remaining time of the execution.



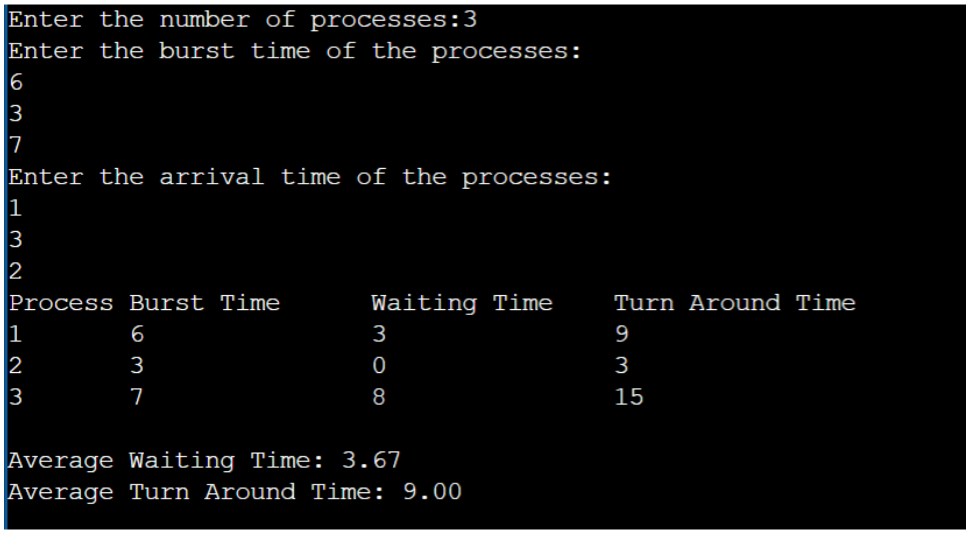
```c
#include<stdlib.h>
#include<stdio.h>
int main(){
    int n;
    printf("Enter the number of processes:");
    scanf("%d", &n);
    int bt[n], wt[n], tat[n], p[n],at[n],btcopy[n];
    printf("Enter the burst time of the processes:\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &bt[i]);
        btcopy[i]=bt[i];
        p[i] = i + 1;
    }
    printf("Enter the arrival time of the processes:\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &at[i]);
    }
    int timeElapsed=0;
    int completed=0;
    while(completed!=n){
        int min=9999;
        int minIndex=-1;
```



```c
        for(int i=0;i<n;i++){
            if(at[i]<=timeElapsed && bt[i]<min && bt[i]>0){
                min=bt[i];
                minIndex=i;
            }
        }
        if(minIndex==-1){
            timeElapsed++;
            continue;
        }
        bt[minIndex]--;
        if(bt[minIndex]==0){
            completed++;
            tat[minIndex]=timeElapsed+1-at[minIndex];
            wt[minIndex]=tat[minIndex]-btcopy[minIndex];
        }
        timeElapsed++;

    }
    float avgwt = 0, avgtat = 0;
    printf("Process\tBurst Time\tWaiting Time\tTurn Around Time");
    printf("\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d\t%d\t\t%d\t\t%d\n", p[i], btcopy[i], wt[i], tat[i]);
        avgwt += wt[i];
        avgtat += tat[i];
    }
    avgwt /= n;
    avgtat /= n;
    printf("\n");
    printf("Average Waiting Time: %.2f\n", avgwt);
    printf("Average Turn Around Time: %.2f\n", avgtat);
}
```

OUTPUT

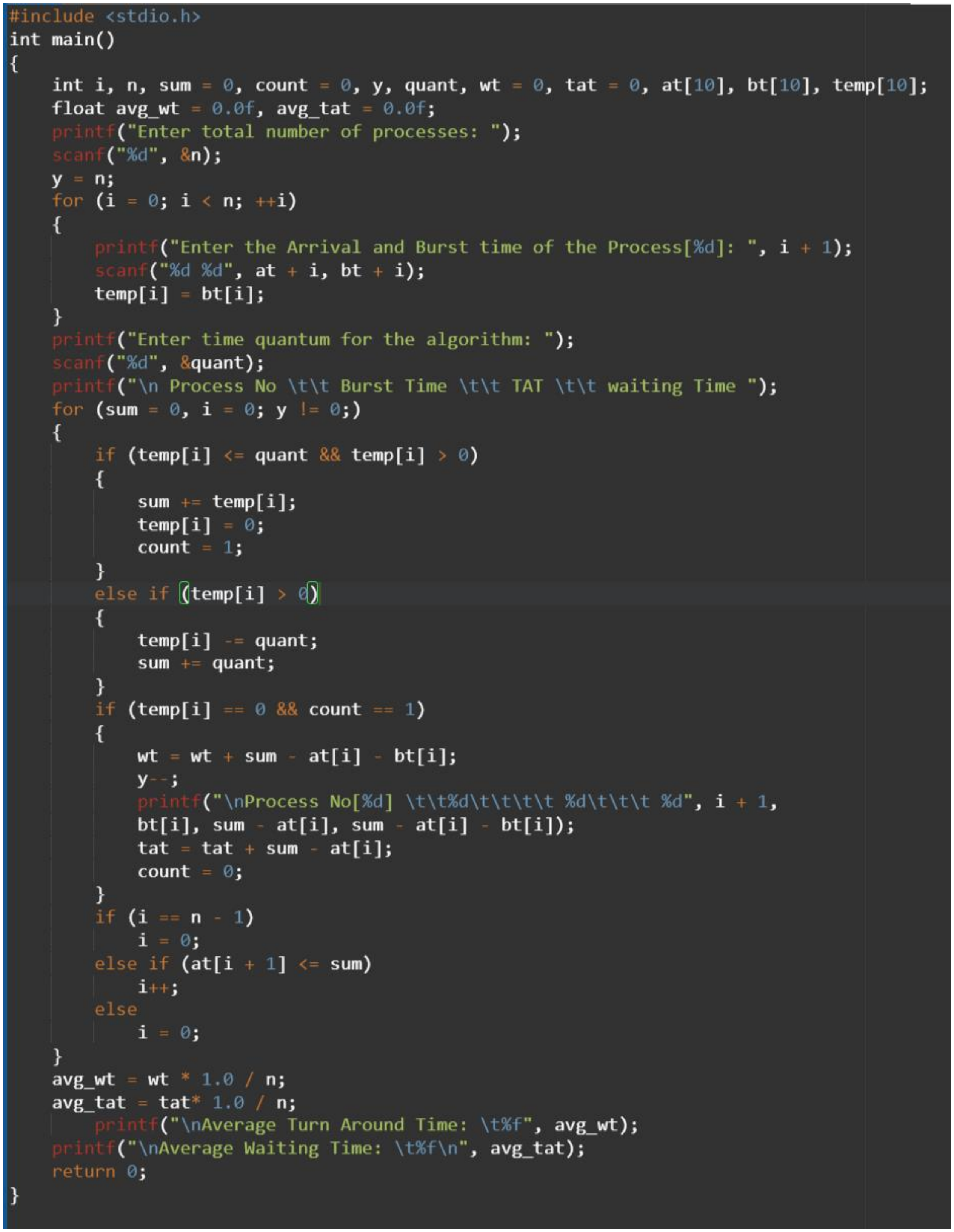


```
Enter the number of processes:3
Enter the burst time of the processes:
6
3
7
Enter the arrival time of the processes:
1
3
2
Process Burst Time      Waiting Time    Turn Around Time
1       6               3               9
2       3               0               3
3       7               8               15

Average Waiting Time: 3.67
Average Turn Around Time: 9.00
```

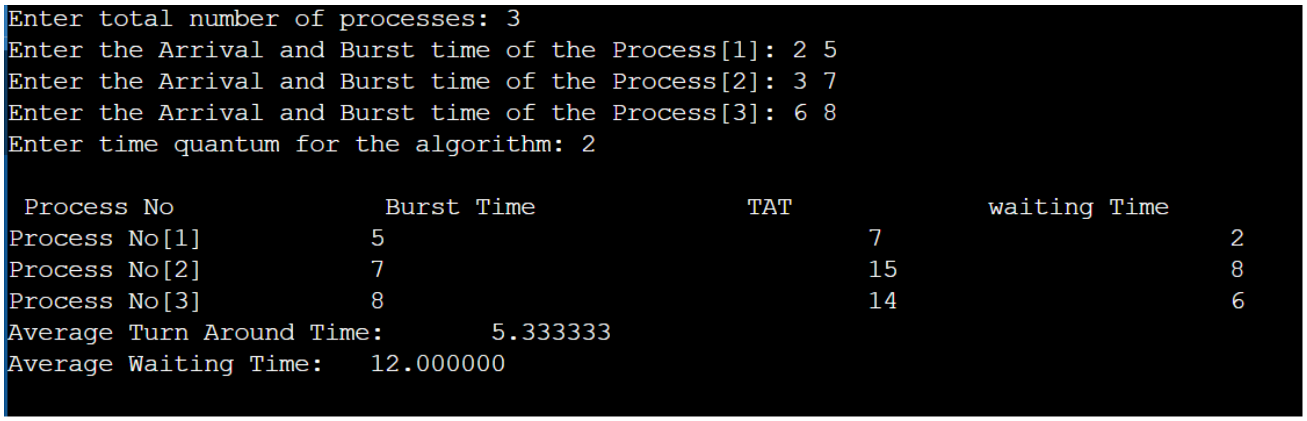
4. Round Robin

In the Round Robin scheduling algorithm, the OS defines a time quantum (slice). All the processes will get executed in the cyclic way. Each of the process will get the CPU for a small amount of time (called time quantum) and then get back to the ready queue to wait for its next turn. It is a preemptive type of scheduling.



```c
#include <stdio.h>
int main()
{
    int i, n, sum = 0, count = 0, y, quant, wt = 0, tat = 0, at[10], bt[10], temp[10];
    float avg_wt = 0.0f, avg_tat = 0.0f;
    printf("Enter total number of processes: ");
    scanf("%d", &n);
    y = n;
    for (i = 0; i < n; ++i)
    {
        printf("Enter the Arrival and Burst time of the Process[%d]: ", i + 1);
        scanf("%d %d", at + i, bt + i);
        temp[i] = bt[i];
    }
    printf("Enter time quantum for the algorithm: ");
    scanf("%d", &quant);
    printf("\n Process No \t\t Burst Time \t\t TAT \t\t waiting Time ");
    for (sum = 0, i = 0; y != 0;)
    {
        if (temp[i] <= quant && temp[i] > 0)
        {
            sum += temp[i];
            temp[i] = 0;
            count = 1;
        }
        else if (temp[i] > 0)
        {
            temp[i] -= quant;
            sum += quant;
        }
        if (temp[i] == 0 && count == 1)
        {
            wt = wt + sum - at[i] - bt[i];
            y--;
            printf("\nProcess No[%d] \t\t%d\t\t\t\t %d\t\t\t %d", i + 1,
            bt[i], sum - at[i], sum - at[i] - bt[i]);
            tat = tat + sum - at[i];
            count = 0;
        }
        if (i == n - 1)
            i = 0;
        else if (at[i + 1] <= sum)
            i++;
        else
            i = 0;
    }
    avg_wt = wt * 1.0 / n;
    avg_tat = tat* 1.0 / n;
        printf("\nAverage Turn Around Time: \t%f", avg_wt);
    printf("\nAverage Waiting Time: \t%f\n", avg_tat);
    return 0;
}
```

OUTPUT-



```
Enter total number of processes: 3
Enter the Arrival and Burst time of the Process[1]: 2 5
Enter the Arrival and Burst time of the Process[2]: 3 7
Enter the Arrival and Burst time of the Process[3]: 6 8
Enter time quantum for the algorithm: 2

 Process No              Burst Time                TAT             waiting Time
Process No[1]           5                                 7                        2
Process No[2]           7                                 15                       8
Process No[3]           8                                 14                       6
Average Turn Around Time:       5.333333
Average Waiting Time:   12.000000
```