3. Round Robin Scheduling

c

#**include**<stdio.h>

**int** main() {

**int** n, i, qt, count=0, temp, sq=0;

**int** bt[10], wt[10], tat[10], rem\_bt[10];

**float** avg\_wt=0, avg\_tat=0;

printf("Enter number of processes: ");

scanf("%d",&n);

printf("Enter Burst Time:\n");

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

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

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

rem\_bt[i] = bt[i];

}

printf("Enter Quantum Time: ");

scanf("%d",&qt);

**while**(1) {

**for**(i=0,count=0;i<n;i++) {

temp = qt;

**if**(rem\_bt[i] == 0) {

count++;

**continue**;

}

**if**(rem\_bt[i] > qt)

rem\_bt[i] -= qt;

**else**

**if**(rem\_bt[i] >= 0) {

temp = rem\_bt[i];

rem\_bt[i] = 0;

}

sq += temp;

tat[i] = sq;

}

**if**(n == count)

**break**;

}

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

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

avg\_wt += wt[i];

avg\_tat += tat[i];

}

printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time");

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

printf("\nP[%d]\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);

}

printf("\n\nAverage Waiting Time: %.2f",avg\_wt/n);

printf("\nAverage Turnaround Time: %.2f",avg\_tat/n);

**return** 0;

}

**Output:**

text

Enter number of processes: 3

Enter Burst Time:

P[1]: 4

P[2]: 5

P[3]: 6

Enter Quantum Time: 2

Process Burst Time Waiting Time Turnaround Time

P[1] 4 9 13

P[2] 5 10 15

P[3] 6 11 17

Average Waiting Time: 10.00

Average Turnaround Time: 15.00

*Implementation following Round Robin algorithm described in Scaler Topics*[*3*](https://www.scaler.com/topics/round-robin-scheduling-program-in-c/)