

/*rr with at*/

```
#include <stdio.h>

int main() {

    int n;

    printf("Enter Total Number of Processes: ");

    scanf("%d", &n);


    int wait_time = 0, ta_time = 0;

    int arrival_time[n], burst_time[n], remaining_burst_time[n];

    int time_slice;

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

        printf("Enter Details of Process %d\n", i + 1);

        printf("Arrival Time: ");

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

        printf("Burst Time: ");

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

        remaining_burst_time[i] = burst_time[i];

    }

    printf("Enter Time Slice (Quantum): ");

    scanf("%d", &time_slice);

    int total_time = 0;

    int completed_processes = 0;

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


    int current_process = 0;

    while (completed_processes < n) {

        if (remaining_burst_time[current_process] > 0) {

            int execution_time;

            if (remaining_burst_time[current_process] > time_slice) {

                execution_time = time_slice;

            } else {
```

```

        execution_time = remaining_burst_time[current_process];
    }

    total_time += execution_time;
    remaining_burst_time[current_process] -= execution_time;

    if (remaining_burst_time[current_process] == 0) {
        completed_processes++;

        int turnaround_time = total_time - arrival_time[current_process];
        int waiting_time = turnaround_time - burst_time[current_process];

        printf("%d\t\t%d\t\t%d\t\t%d\n", current_process + 1, burst_time[current_process],
            turnaround_time, waiting_time);

        wait_time += waiting_time;
        ta_time += turnaround_time;
    }

    // Move to the next process in a circular manner
    current_process = (current_process + 1) % n;
} else {
    // If the process has already completed, move to the next process
    current_process = (current_process + 1) % n;
}
}

float average_wait_time = (float)wait_time / n;
float average_turnaround_time = (float)ta_time / n;

printf("\nAverage Waiting Time: %f", average_wait_time);

```

```
printf("\nAverage Turnaround Time: %f\n", average_turnaround_time);
```

```
return 0;
```

```
}
```

The screenshot shows an online C compiler interface with the following components:

- Source Code:** A C program implementing Round Robin scheduling. It prompts the user for the number of processes, arrival times, and burst times. It then calculates the average waiting and turnaround times.
- Terminal:** Displays the program's output, including the input values and the calculated average times.
- Table:** A table showing the execution details for each process.

```
1  /*rr with at*/
2  #include <stdio.h>
3  int main() {
4      int n;
5      printf("Enter Total Number of Processes: ");
6      scanf("%d", &n);
7
8      int wait_time = 0, ta_time = 0;
9      int arrival_time[n], burst_time[n], remaining_burst_time[n];
10     int time_slice;
11     for (int i = 0; i < n; i++) {
12         printf("Enter Details of Process %d\n", i + 1);
13         printf("Arrival Time: ");
14         scanf("%d", &arrival_time[i]);
15         printf("Burst Time: ");
16         scanf("%d", &burst_time[i]);
17         remaining_burst_time[i] = burst_time[i];
18     }
19     printf("Enter Time Slice (Quantum): ");
20     scanf("%d", &time_slice);
21     int total_time = 0;
22     int completed_processes = 0;
23     printf("\nProcess ID\tBurst Time\tTurnaround Time\tWaiting Time\n");
24
25     int current_process = 0;
26     while (completed_processes < n) {
27         if (remaining_burst_time[current_process] > 0) {
28             int execution_time;
29             if (remaining_burst_time[current_process] > time_slice) {
```

Enter Total Number of Processes: 3
Enter Details of Process 1
Arrival Time: 2
Burst Time: 24
Enter Details of Process 2
Arrival Time: 0
Burst Time: 4
Enter Details of Process 3
Arrival Time: 1
Burst Time: 3
Enter Time Slice (Quantum): 4

Process ID	Burst Time	Turnaround Time	Waiting Time
2	4	8	4
3	3	10	7
1	24	29	5

Average Waiting Time: 5.333333
Average Turnaround Time: 15.666667