

Name: K.SaiKrishna

Reg-No: 192311106

8. Construct a C program to simulate Round Robin scheduling algorithm with C.

Aim

To simulate the Round Robin CPU scheduling algorithm using C, demonstrating how processes share the CPU time in a fair and cyclic manner.

Algorithm

1. Input the number of processes, their burst times, and the time quantum.
2. Initialize a queue to maintain process order and track remaining burst times.
3. Execute each process for the time quantum or until it finishes, whichever comes first.
4. Update remaining burst times and re-add processes to the queue if not completed.
5. Repeat until all processes finish.
6. Calculate and display turnaround time and waiting time for each process.

Procedure

1. Read the number of processes, burst times, and time quantum.
2. Simulate process execution by iterating through the queue cyclically, decrementing the burst time.
3. Track the time at which processes finish to calculate their turnaround and waiting times.
4. Output the results.

Code:

```
#include <stdio.h>
```

```
void roundRobin(int n, int burst_time[], int quantum) {
```

```
    int remaining_bt[n], wait_time[n], turn_time[n], total_wait = 0, total_turn = 0;
```

```
    for (int i = 0; i < n; i++) remaining_bt[i] = burst_time[i];
```

```
    int time = 0, completed = 0;
```

```
    while (completed < n) {
```

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

```

    if (remaining_bt[i] > 0) {

        if (remaining_bt[i] > quantum) {

            time += quantum;

            remaining_bt[i] -= quantum;

        } else {

            time += remaining_bt[i];

            wait_time[i] = time - burst_time[i];

            remaining_bt[i] = 0;

            turn_time[i] = time;

            completed++;

        }

    }

}

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

    total_wait += wait_time[i];

    total_turn += turn_time[i];

}

printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");

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

    printf("P%d\t%d\t%d\t%d\n", i + 1, burst_time[i], wait_time[i], turn_time[i]);

```

```
}
```

```
printf("Average Waiting Time: %.2f\n", (float)total_wait / n);
```

```
printf("Average Turnaround Time: %.2f\n", (float)total_turn / n);
```

```
}
```

```
int main() {
```

```
    int n, quantum;
```

```
    printf("Enter the number of processes: ");
```

```
    scanf("%d", &n);
```

```
    int burst_time[n];
```

```
    printf("Enter the burst time for each process:\n");
```

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

```
        printf("Process P%d: ", i + 1);
```

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

```
    }
```

```
    printf("Enter the time quantum: ");
```

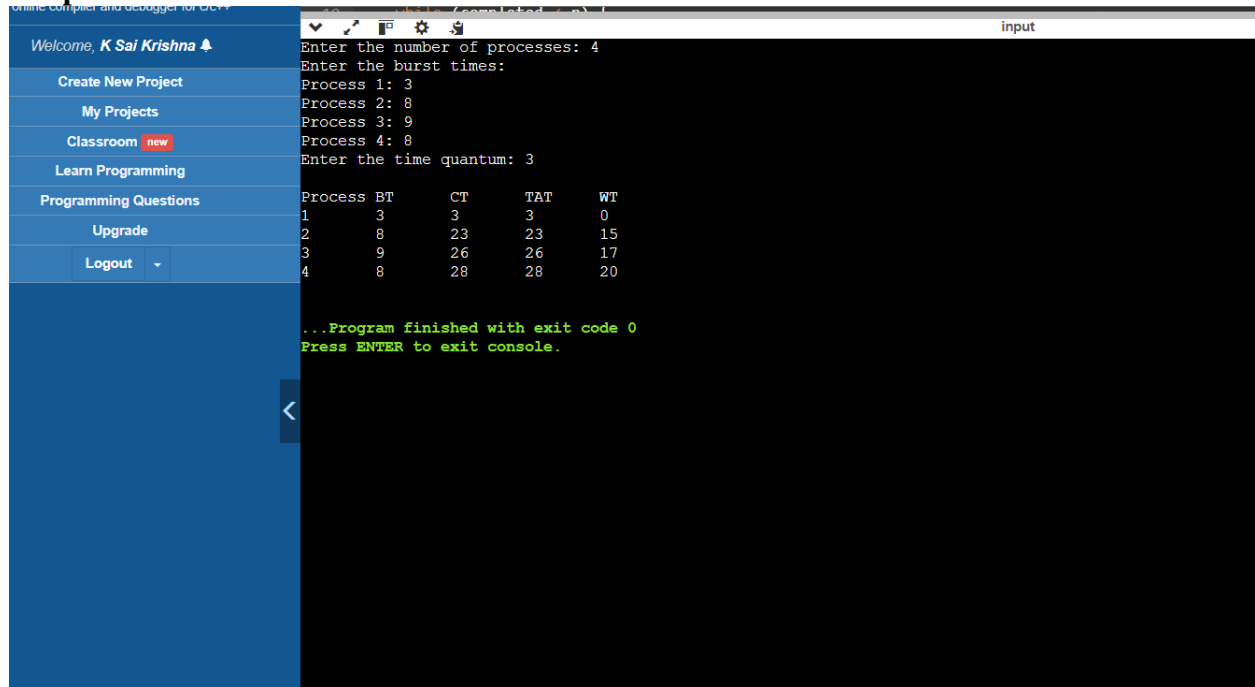
```
    scanf("%d", &quantum);
```

```
    roundRobin(n, burst_time, quantum);
```

```
    return 0;
```

}

Output:



```
Online Compiler and Debugger for C/C++
Welcome, K Sai Krishna
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Upgrade
Logout
Enter the number of processes: 4
Enter the burst times:
Process 1: 3
Process 2: 8
Process 3: 9
Process 4: 8
Enter the time quantum: 3
Process BT      CT      TAT      WT
1      3      3      3      0
2      8      23     23     15
3      9      26     26     17
4      8      28     28     20
...Program finished with exit code 0
Press ENTER to exit console.
```

Result

For a set of inputs (e.g., 3 processes with burst times 10, 5, and 8, and time quantum = 4):