OS LAB WEEK 3

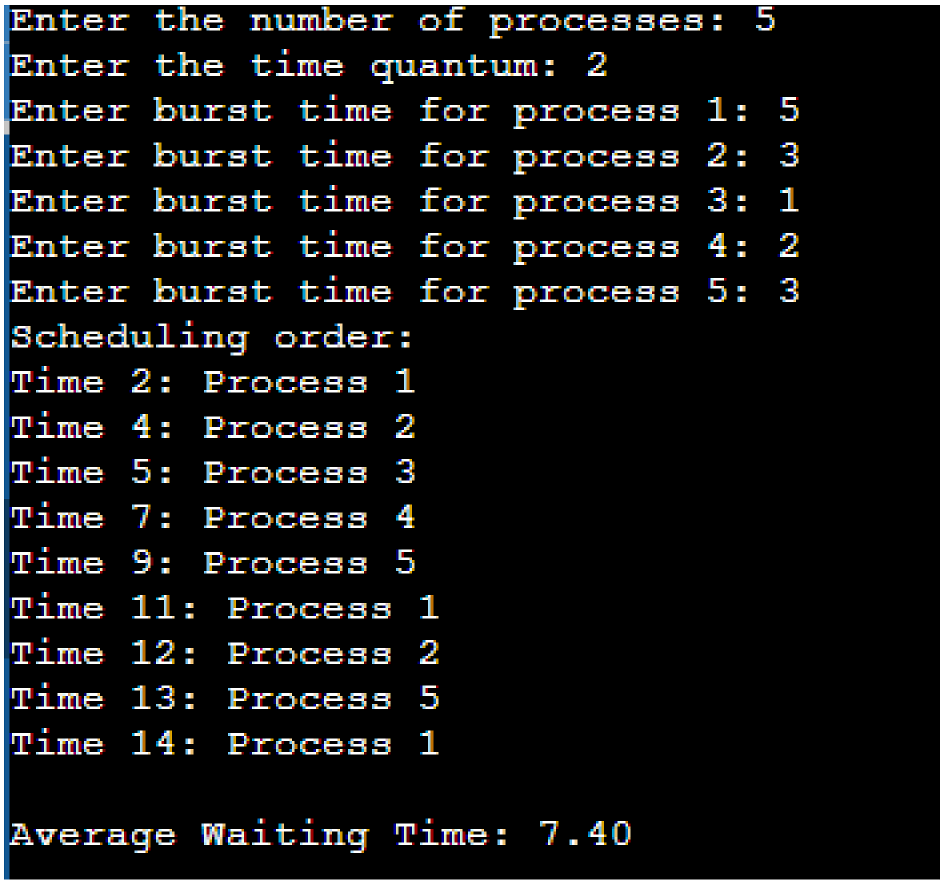
Yash Gupta
1BM21CS251
21/06/23

Q: Round Robin using C

```
#include <stdio.h>
#define MAX_SIZE 100
struct Process {
    int pid;
    int burst_time;
    int remaining_time;
    int waiting_time;
};

void RoundRobin(struct Process processes[], int n, int time_quantum) {
    int total_time = 0;
    int completed = 0;
    int ready_queue[MAX_SIZE];
    int front = 0, rear = -1;
    for (int i = 0; i < n; i++) {
        ready_queue[++rear] = i;
    }

    while (completed < n) {
        int current_process = ready_queue[front++];
        if (processes[current_process].remaining_time > 0) {
            if (processes[current_process].remaining_time <= time_quantum) {
                total_time += processes[current_process].remaining_time;
                processes[current_process].remaining_time = 0;
            } else {
                total_time += time_quantum;
                processes[current_process].remaining_time -= time_quantum;
            }

            printf("Time %d: Process %d\n", total_time,
processes[current_process].pid);
        }
        if (processes[current_process].remaining_time == 0) {
            completed++;
            processes[current_process].waiting_time = total_time -
processes[current_process].burst_time;
        } else {
            ready_queue[++rear] = current_process;
        }
    }
}

int main() {
    int n;
    int time_quantum;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter the time quantum: ");
    scanf("%d", &time_quantum);
    struct Process processes[n];
    for (int i = 0; i < n; i++) {
        printf("Enter burst time for process %d: ", i + 1);
        scanf("%d", &processes[i].burst_time);
        processes[i].remaining_time = processes[i].burst_time;
        processes[i].pid = i + 1;
    }
    printf("Scheduling order:\n");
    RoundRobin(processes, n, time_quantum);
```

```
    double total_waiting_time = 0;
    for (int i = 0; i < n; i++) {
        total_waiting_time += processes[i].waiting_time;
    }
    double avg_waiting_time = total_waiting_time / n;
    printf("\nAverage Waiting Time: %.2lf\n", avg_waiting_time);
    return 0;
}
```

Output



```
Enter the number of processes: 5
Enter the time quantum: 2
Enter burst time for process 1: 5
Enter burst time for process 2: 3
Enter burst time for process 3: 1
Enter burst time for process 4: 2
Enter burst time for process 5: 3
Scheduling order:
Time 2: Process 1
Time 4: Process 2
Time 5: Process 3
Time 7: Process 4
Time 9: Process 5
Time 11: Process 1
Time 12: Process 2
Time 13: Process 5
Time 14: Process 1

Average Waiting Time: 7.40
```