NAME: ARUN MC

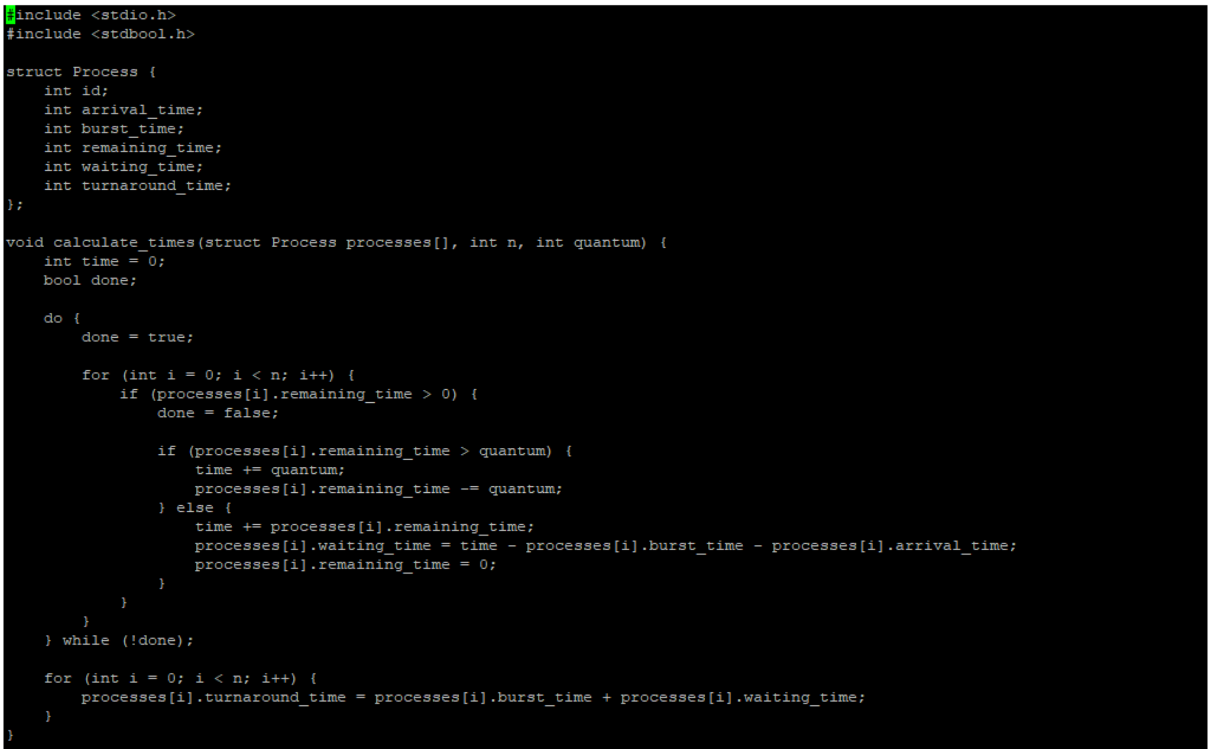
ROLL NO: 230701036

Exp:6d **ROUND ROBIN SCHEDULING**

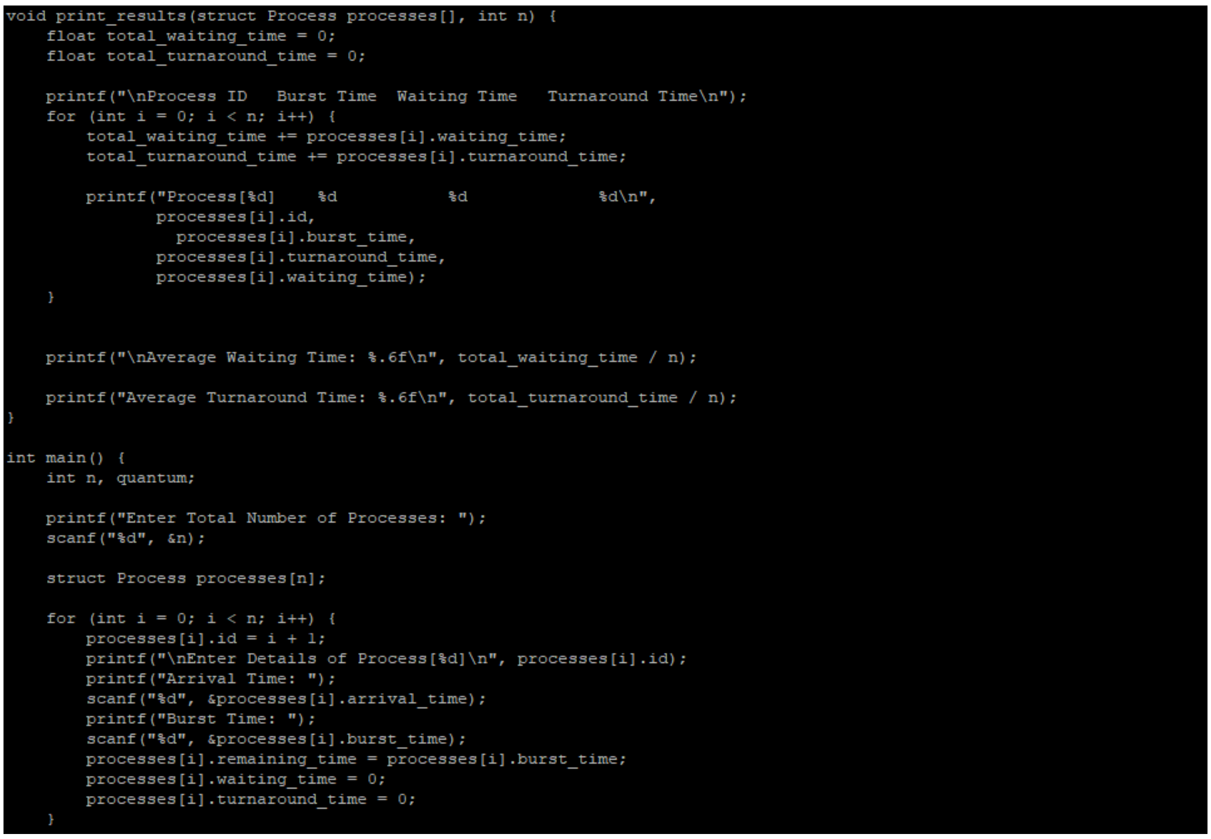
Aim:

To implement the Round Robin (RR) scheduling technique

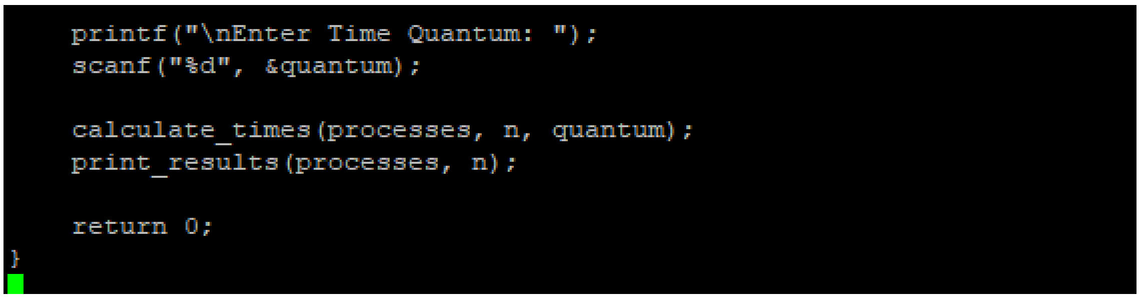
CODE:



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

struct Process {
    int id;
    int arrival_time;
    int burst_time;
    int remaining_time;
    int waiting_time;
    int turnaround_time;
};

void calculate_times(struct Process processes[], int n, int quantum) {
    int time = 0;
    bool done;

    do {
        done = true;

        for (int i = 0; i < n; i++) {
            if (processes[i].remaining_time > 0) {
                done = false;

                if (processes[i].remaining_time > quantum) {
                    time += quantum;
                    processes[i].remaining_time -= quantum;
                } else {
                    time += processes[i].remaining_time;
                    processes[i].waiting_time = time - processes[i].burst_time - processes[i].arrival_time;
                    processes[i].remaining_time = 0;
                }
            }
        }
    } while (!done);

    for (int i = 0; i < n; i++) {
        processes[i].turnaround_time = processes[i].burst_time + processes[i].waiting_time;
    }
}
```

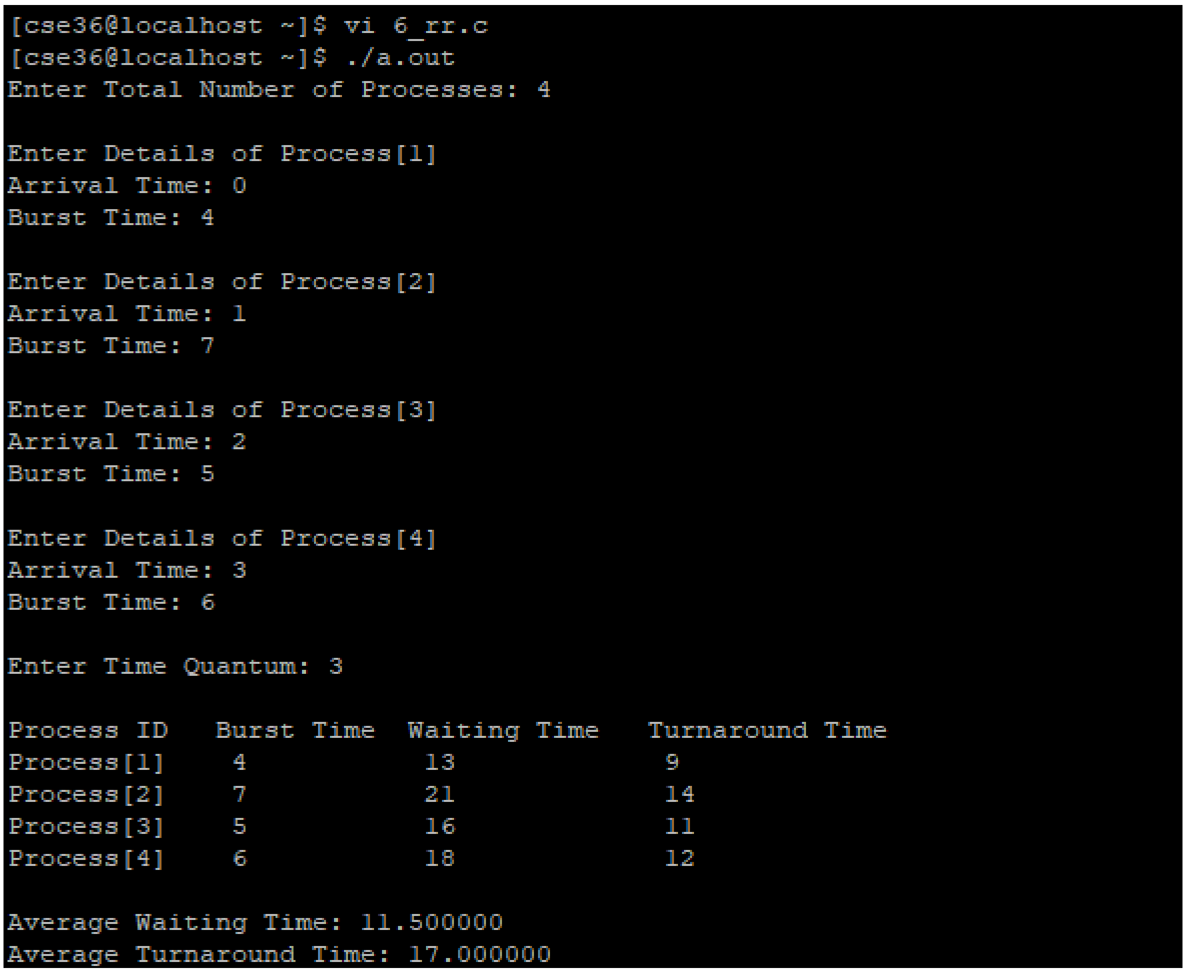


```
void print_results(struct Process processes[], int n) {
    float total_waiting_time = 0;
    float total_turnaround_time = 0;

    printf("\nProcess ID   Burst Time  Waiting Time   Turnaround Time\n");
    for (int i = 0; i < n; i++) {
        total_waiting_time += processes[i].waiting_time;
        total_turnaround_time += processes[i].turnaround_time;

        printf("Process[%d]    %d           %d             %d\n",
               processes[i].id,
                 processes[i].burst_time,
               processes[i].turnaround_time,
               processes[i].waiting_time);
    }


    printf("\nAverage Waiting Time: %.6f\n", total_waiting_time / n);

    printf("Average Turnaround Time: %.6f\n", total_turnaround_time / n);
}

int main() {
    int n, quantum;

    printf("Enter Total Number of Processes: ");
    scanf("%d", &n);

    struct Process processes[n];

    for (int i = 0; i < n; i++) {
        processes[i].id = i + 1;
        printf("\nEnter Details of Process[%d]\n", processes[i].id);
        printf("Arrival Time: ");
        scanf("%d", &processes[i].arrival_time);
        printf("Burst Time: ");
        scanf("%d", &processes[i].burst_time);
        processes[i].remaining_time = processes[i].burst_time;
        processes[i].waiting_time = 0;
        processes[i].turnaround_time = 0;
    }
```



```
    printf("\nEnter Time Quantum: ");
    scanf("%d", &quantum);

    calculate_times(processes, n, quantum);
    print_results(processes, n);

    return 0;
}
```

OUTPUT:


```
[cse36@localhost ~]$ vi 6_rr.c
[cse36@localhost ~]$ ./a.out
Enter Total Number of Processes: 4

Enter Details of Process[1]
Arrival Time: 0
Burst Time: 4

Enter Details of Process[2]
Arrival Time: 1
Burst Time: 7

Enter Details of Process[3]
Arrival Time: 2
Burst Time: 5

Enter Details of Process[4]
Arrival Time: 3
Burst Time: 6

Enter Time Quantum: 3

Process ID   Burst Time  Waiting Time   Turnaround Time
Process[1]    4           13             9
Process[2]    7           21             14
Process[3]    5           16             11
Process[4]    6           18             12

Average Waiting Time: 11.500000
Average Turnaround Time: 17.000000
```