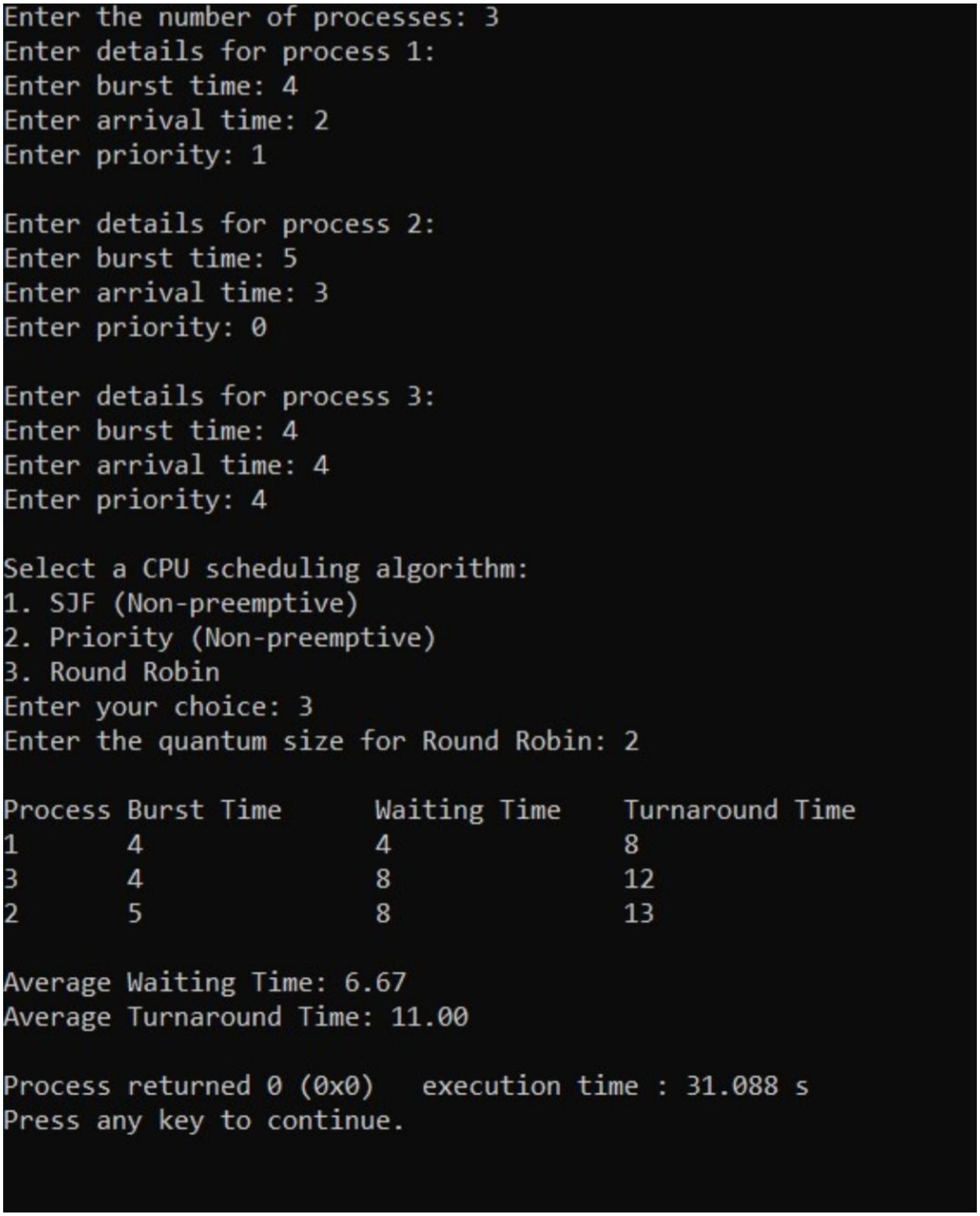
USN :1BM21CS222

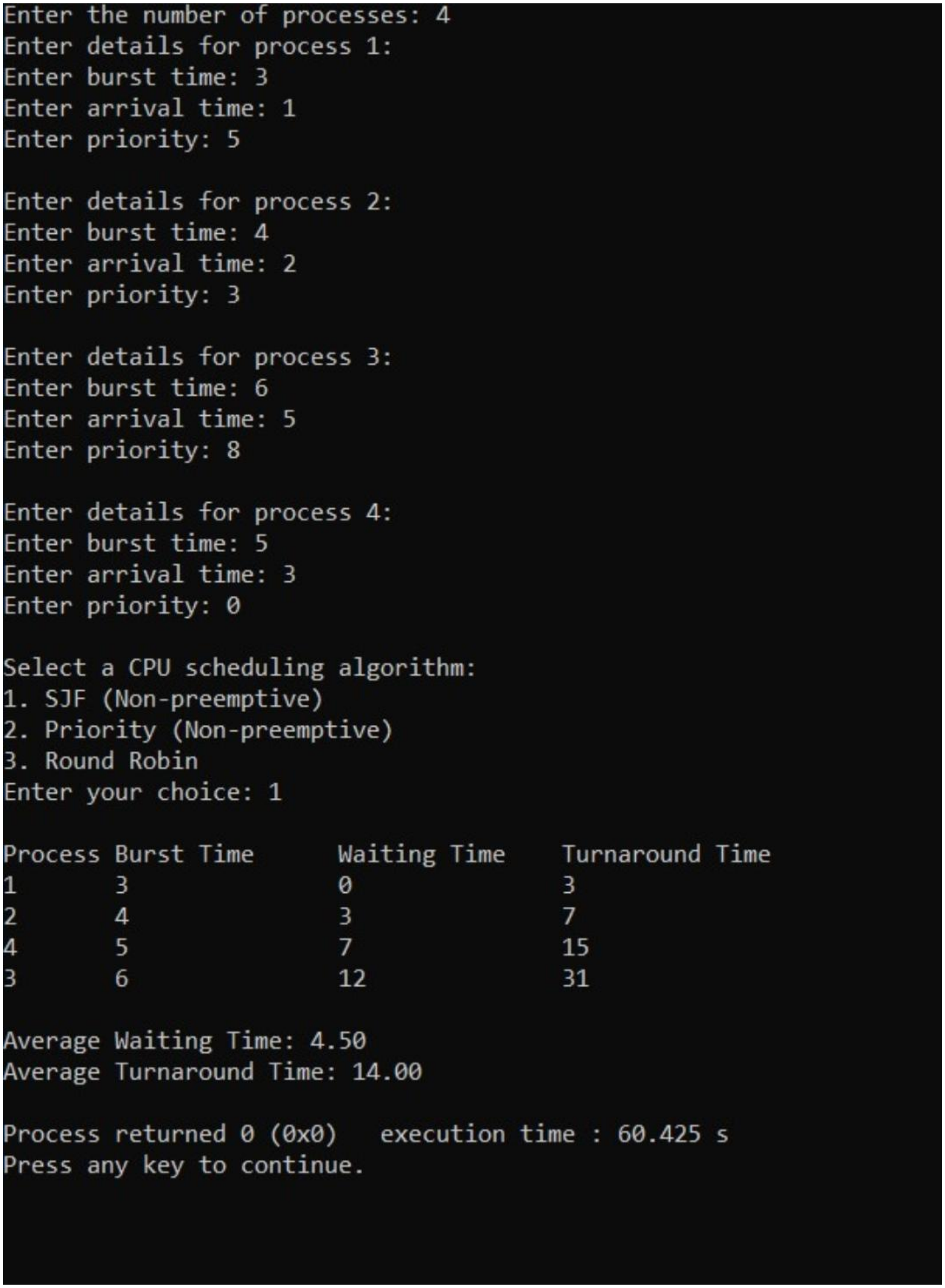
CODE:

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

struct Process {
    int pid;
    int burstTime;
    int arrivalTime;
    int priority;
    int waitingTime;
    int turnaroundTime;
    int remainingTime;
};

void sjfNonPreemptive(struct Process processes[], int n);
void priorityNonPreemptive(struct Process processes[], int n);
void roundRobin(struct Process processes[], int n, int quantum);

int main() {
    int n, choice, quantum;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    struct Process processes[n];

    for (int i = 0; i < n; i++) {
        printf("Enter details for process %d:\n", i + 1);
        printf("Enter burst time: ");
        scanf("%d", &processes[i].burstTime);
        printf("Enter arrival time: ");
        scanf("%d", &processes[i].arrivalTime);
        printf("Enter priority: ");
        scanf("%d", &processes[i].priority);
        processes[i].pid = i + 1;
        processes[i].waitingTime = 0;
        processes[i].turnaroundTime = 0;
        processes[i].remainingTime = processes[i].burstTime;
        printf("\n");
    }

    printf("Select a CPU scheduling algorithm:\n");
    printf("1. SJF (Non-preemptive)\n");
    printf("2. Priority (Non-preemptive)\n");
    printf("3. Round Robin\n");
    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            sjfNonPreemptive(processes, n);
            break;
        case 2:
            priorityNonPreemptive(processes, n);
            break;
        case 3:
            printf("Enter the quantum size for Round Robin: ");
            scanf("%d", &quantum);
            roundRobin(processes, n, quantum);
            break;
        default:
            printf("Invalid choice.\n");
            break;
    }

    return 0;
}

void sjfNonPreemptive(struct Process processes[], int n) {

    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (processes[j].burstTime > processes[j + 1].burstTime) {
                struct Process temp = processes[j];
                processes[j] = processes[j + 1];
                processes[j + 1] = temp;
            }
        }
    }

    int totalWaitingTime = 0;
    int totalTurnaroundTime = 0;

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

    for (int i = 0; i < n; i++) {
        processes[i].waitingTime = totalWaitingTime;

        processes[i].turnaroundTime = totalTurnaroundTime + processes[i].burstTime;

        totalWaitingTime += processes[i].burstTime;

        totalTurnaroundTime += processes[i].turnaroundTime;


        printf("%d\t%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].burstTime,

               processes[i].waitingTime, processes[i].turnaroundTime);

    }


    double averageWaitingTime = (double)totalWaitingTime / n;

    double averageTurnaroundTime = (double)totalTurnaroundTime / n;


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

    printf("Average Turnaround Time: %.2f\n", averageTurnaroundTime);

}


void priorityNonPreemptive(struct Process processes[], int n) {


    for (int i = 0; i < n - 1; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (processes[j].priority > processes[j + 1].priority) {

                struct Process temp = processes[j];

                processes[j] = processes[j + 1];

                processes[j + 1] = temp;

            }

        }

    }
    int totalWaitingTime = 0;
    int totalTurnaroundTime = 0;

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

    for (int i = 0; i < n; i++) {
        processes[i].waitingTime = totalWaitingTime;
        processes[i].turnaroundTime = totalTurnaroundTime + processes[i].burstTime;
        totalWaitingTime += processes[i].burstTime;
        totalTurnaroundTime += processes[i].turnaroundTime;

        printf("%d\t%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].burstTime,
               processes[i].waitingTime, processes[i].turnaroundTime);
    }

    double averageWaitingTime = (double)totalWaitingTime / n;
    double averageTurnaroundTime = (double)totalTurnaroundTime / n;

    printf("\nAverage Waiting Time: %.2f\n", averageWaitingTime);
    printf("Average Turnaround Time: %.2f\n", averageTurnaroundTime);
}

void roundRobin(struct Process processes[], int n, int quantum) {

    int totalWaitingTime = 0;
    int totalTurnaroundTime = 0;
    int remainingProcesses = n;
    int currentTime = 0;
    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");

    while (remainingProcesses > 0) {
        for (int i = 0; i < n; i++) {
            if (processes[i].remainingTime <= quantum && processes[i].remainingTime > 0) {
                currentTime += processes[i].remainingTime;
                processes[i].remainingTime = 0;
                processes[i].waitingTime = currentTime - processes[i].burstTime;
                processes[i].turnaroundTime = currentTime;
                totalWaitingTime += processes[i].waitingTime;
                totalTurnaroundTime += processes[i].turnaroundTime;
                remainingProcesses--;

                printf("%d\t%d\t\t%d\t\t%d\n", processes[i].pid, processes[i].burstTime,
                       processes[i].waitingTime, processes[i].turnaroundTime);
            } else if (processes[i].remainingTime > 0) {
                currentTime += quantum;
                processes[i].remainingTime -= quantum;
            }
        }
    }

    double averageWaitingTime = (double)totalWaitingTime / n;
    double averageTurnaroundTime = (double)totalTurnaroundTime / n;

    printf("\nAverage Waiting Time: %.2f\n", averageWaitingTime);
    printf("Average Turnaround Time: %.2f\n", averageTurnaroundTime);
}
```

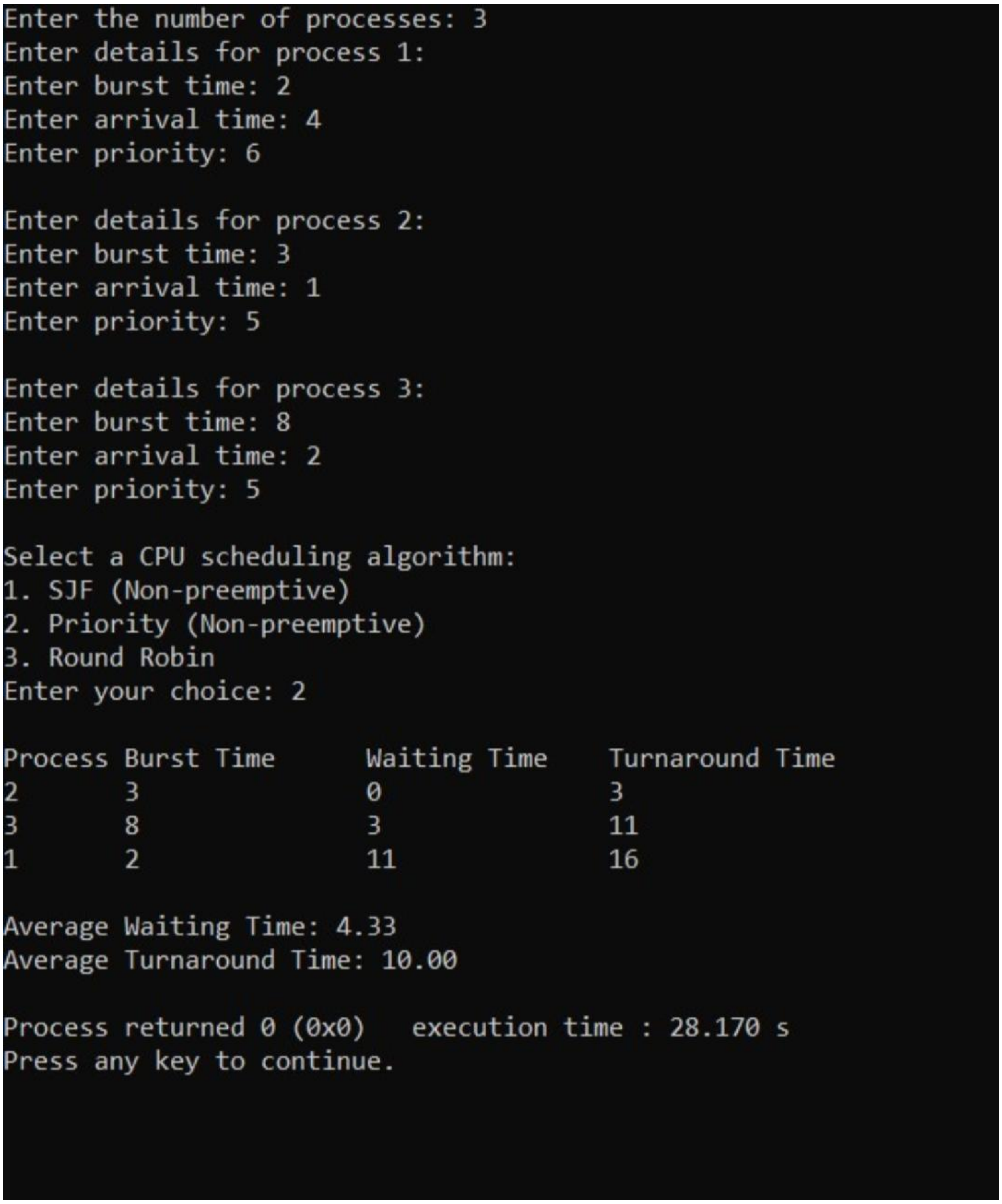
Output:



```
Enter the number of processes: 3
Enter details for process 1:
Enter burst time: 4
Enter arrival time: 2
Enter priority: 1

Enter details for process 2:
Enter burst time: 5
Enter arrival time: 3
Enter priority: 0

Enter details for process 3:
Enter burst time: 4
Enter arrival time: 4
Enter priority: 4

Select a CPU scheduling algorithm:
1. SJF (Non-preemptive)
2. Priority (Non-preemptive)
3. Round Robin
Enter your choice: 3
Enter the quantum size for Round Robin: 2

Process Burst Time      Waiting Time    Turnaround Time
1       4               4               8
3       4               8               12
2       5               8               13

Average Waiting Time: 6.67
Average Turnaround Time: 11.00

Process returned 0 (0x0)   execution time : 31.088 s
Press any key to continue.
```



```
Enter the number of processes: 4
Enter details for process 1:
Enter burst time: 3
Enter arrival time: 1
Enter priority: 5

Enter details for process 2:
Enter burst time: 4
Enter arrival time: 2
Enter priority: 3

Enter details for process 3:
Enter burst time: 6
Enter arrival time: 5
Enter priority: 8

Enter details for process 4:
Enter burst time: 5
Enter arrival time: 3
Enter priority: 0

Select a CPU scheduling algorithm:
1. SJF (Non-preemptive)
2. Priority (Non-preemptive)
3. Round Robin
Enter your choice: 1

Process Burst Time      Waiting Time    Turnaround Time
1       3               0               3
2       4               3               7
4       5               7               15
3       6               12              31

Average Waiting Time: 4.50
Average Turnaround Time: 14.00

Process returned 0 (0x0)   execution time : 60.425 s
Press any key to continue.
```



```
Enter the number of processes: 3
Enter details for process 1:
Enter burst time: 2
Enter arrival time: 4
Enter priority: 6

Enter details for process 2:
Enter burst time: 3
Enter arrival time: 1
Enter priority: 5

Enter details for process 3:
Enter burst time: 8
Enter arrival time: 2
Enter priority: 5

Select a CPU scheduling algorithm:
1. SJF (Non-preemptive)
2. Priority (Non-preemptive)
3. Round Robin
Enter your choice: 2

Process Burst Time      Waiting Time    Turnaround Time
2       3               0               3
3       8               3               11
1       2               11              16

Average Waiting Time: 4.33
Average Turnaround Time: 10.00

Process returned 0 (0x0)   execution time : 28.170 s
Press any key to continue.
```