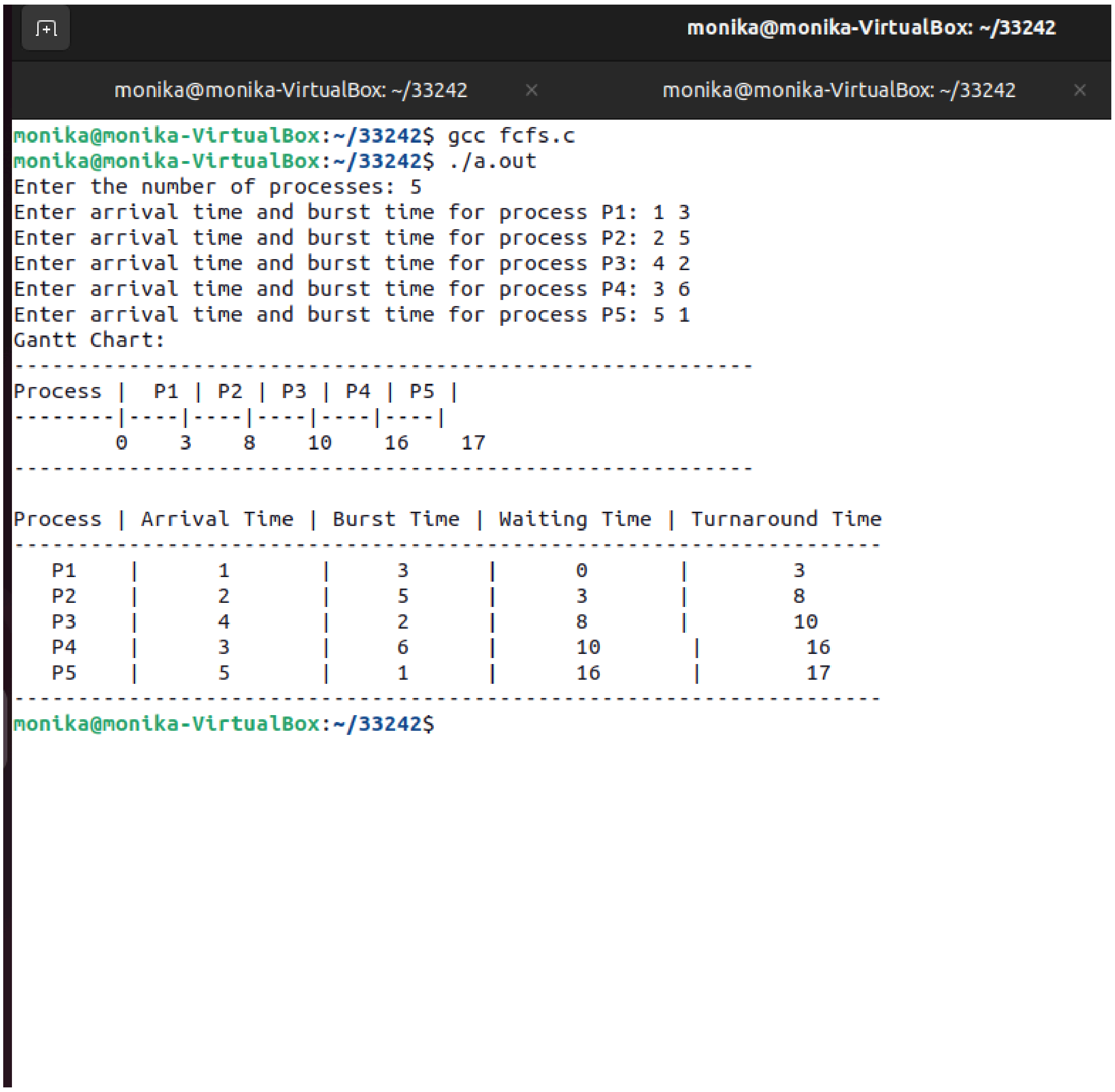
OS Assignment 3 - FCFS , SJF(Preemptive & Non-Preemptive) , Round Robin, Priority
Roll no. 33242

Code for FCFS:

```
#include <stdio.h>

void findWaitingTime(int processes[], int n, int bt[], int wt[], int
at[]) {
    wt[0] = 0; // Waiting time for the first process is 0

    for (int i = 1; i < n; i++) {
        wt[i] = bt[i - 1] + wt[i - 1];
        if (wt[i] < at[i])
            wt[i] = at[i] - wt[i]; // Waiting time cannot be negative
if arrival time is greater
    }
}

void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int
tat[]) {
    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];
}

void findavgTime(int processes[], int n, int bt[], int at[]) {
    int wt[n], tat[n];

    findWaitingTime(processes, n, bt, wt, at);
    findTurnAroundTime(processes, n, bt, wt, tat);

    // Print the Gantt chart
    printf("Gantt Chart:\n");

printf("---------------------------------------------------------\n");
    printf("Process | ");

    for (int i = 0; i < n; i++)
        printf(" P%d |", processes[i]);
    printf("\n");
    printf("--------|");

    for (int i = 0; i < n; i++)
        printf("----|");
    printf("\n");

    printf("        0");
    for (int i = 0; i < n; i++)
        printf("    %d", tat[i]);
    printf("\n");
printf("----------------------------------------------------------\n");

    // Print table with results
    printf("\nProcess | Arrival Time | Burst Time | Waiting Time |
Turnaround Time\n");

printf("--------------------------------------------------------------
-----\n");
    for (int i = 0; i < n; i++) {
        printf("   P%d    |      %d       |     %d      |      %d
|        %d\n",
               processes[i], at[i], bt[i], wt[i], tat[i]);
    }

printf("--------------------------------------------------------------
-----\n");
}

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

    int processes[n], bt[n], at[n];

    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;
        printf("Enter arrival time and burst time for process P%d: ",
processes[i]);
        scanf("%d %d", &at[i], &bt[i]);
    }

    findavgTime(processes, n, bt, at);

    return 0;
}
```

OUTPUT -


monika@monika-VirtualBox: ~/33242
monika@monika-VirtualBox: ~/33242
monika@monika-VirtualBox: ~/33242

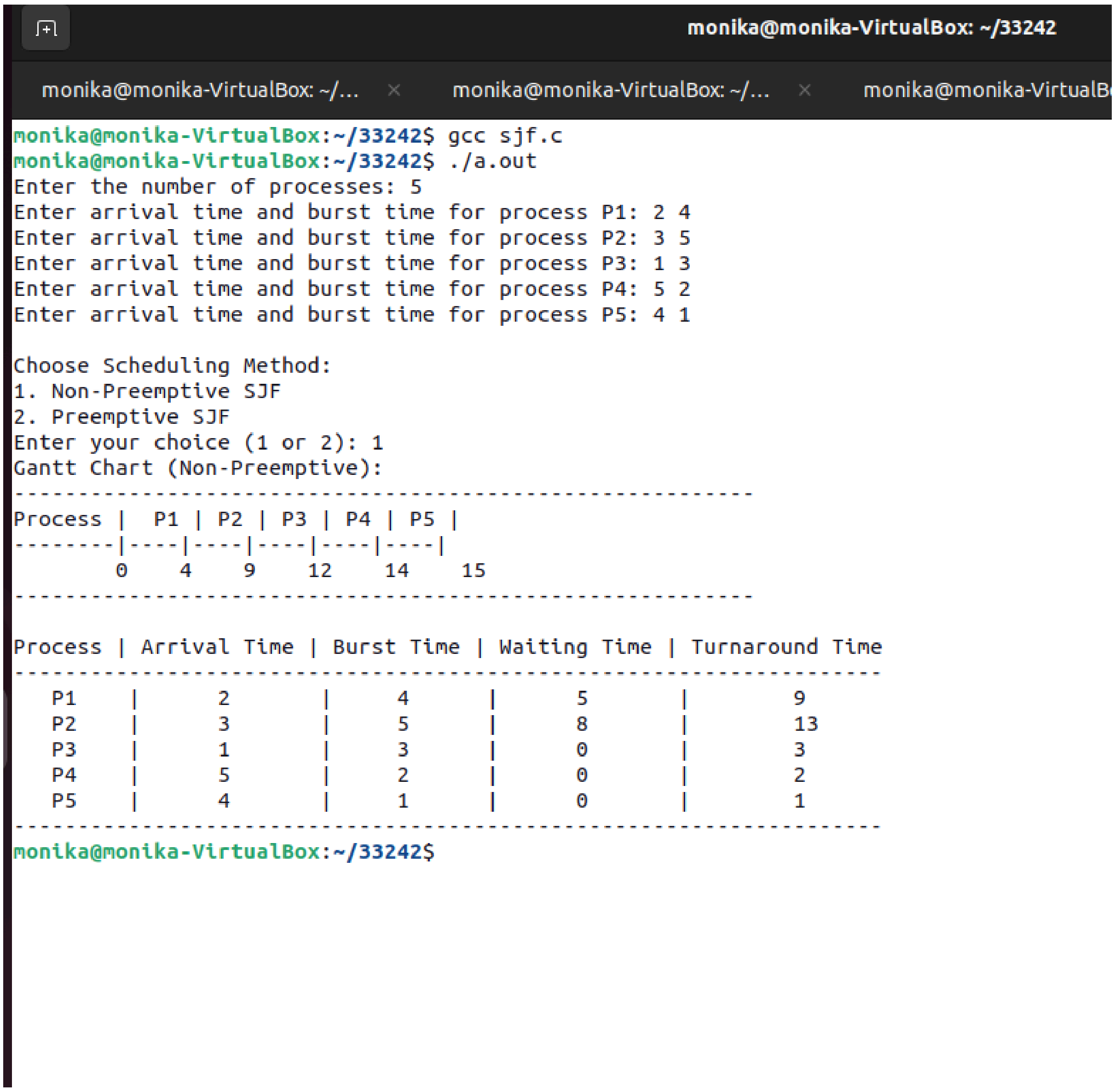
```
monika@monika-VirtualBox:~/33242$ gcc fcfs.c
monika@monika-VirtualBox:~/33242$ ./a.out
Enter the number of processes: 5
Enter arrival time and burst time for process P1: 1 3
Enter arrival time and burst time for process P2: 2 5
Enter arrival time and burst time for process P3: 4 2
Enter arrival time and burst time for process P4: 3 6
Enter arrival time and burst time for process P5: 5 1
Gantt Chart:
-----------------------------------------------------------
Process |  P1 | P2 | P3 | P4 | P5 |
--------|----|----|----|----|----|
        0    3    8    10    16    17
-----------------------------------------------------------

Process | Arrival Time | Burst Time | Waiting Time | Turnaround Time
---------------------------------------------------------------------
   P1    |       1       |      3      |      0       |        3
   P2    |       2       |      5      |      3       |        8
   P3    |       4       |      2      |      8       |        10
   P4    |       3       |      6      |      10       |        16
   P5    |       5       |      1      |      16       |        17
---------------------------------------------------------------------
monika@monika-VirtualBox:~/33242$
```

Code for SJF -

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

void findWaitingTimeNonPreemptive(int processes[], int n, int bt[], int
wt[], int at[]) {
    int completed[n];
    int remaining = n;
    int time = 0;

    for (int i = 0; i < n; i++) {
        completed[i] = 0;
        wt[i] = 0;
    }

    while (remaining > 0) {
        int min = INT_MAX;
        int idx = -1;

        // Find the process with the shortest burst time that has
arrived
        for (int i = 0; i < n; i++) {
            if (at[i] <= time && !completed[i] && bt[i] < min) {
                min = bt[i];
                idx = i;
            }
        }

        if (idx == -1) {
            time++;
            continue;
        }

        wt[idx] = time - at[idx];
        time += bt[idx];
        completed[idx] = 1;
        remaining--;
    }
}

void findTurnAroundTimeNonPreemptive(int processes[], int n, int bt[],
int wt[], int tat[]) {
    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];
}

void printGanttChartNonPreemptive(int processes[], int n, int bt[], int
at[]) {
    int wt[n], tat[n];
    findWaitingTimeNonPreemptive(processes, n, bt, wt, at);
    findTurnAroundTimeNonPreemptive(processes, n, bt, wt, tat);

    // Print the Gantt chart
    printf("Gantt Chart (Non-Preemptive):\n");

printf("----------------------------------------------------------\n");
    printf("Process | ");

    for (int i = 0; i < n; i++)
        printf(" P%d |", processes[i]);
    printf("\n");
    printf("--------|");

    for (int i = 0; i < n; i++)
        printf("----|");
    printf("\n");

    int time = 0;
    printf("        %d", time);
    for (int i = 0; i < n; i++) {
        time += bt[i];
        printf("    %d", time);
    }
    printf("\n");

printf("----------------------------------------------------------\n");

    // Print table with results
    printf("\nProcess | Arrival Time | Burst Time | Waiting Time |
Turnaround Time\n");

printf("----------------------------------------------------------------
-----\n");
    for (int i = 0; i < n; i++) {
        printf("   P%d    |      %d       |     %d      |      %d
|        %d\n",
               processes[i], at[i], bt[i], wt[i], tat[i]);
    }

printf("----------------------------------------------------------------
-----\n");
}

void findWaitingTimePreemptive(int processes[], int n, int bt[], int
wt[], int at[]) {
    int remainingTime[n], completed[n];
    int time = 0, remaining = n;

    for (int i = 0; i < n; i++) {
        remainingTime[i] = bt[i];
        completed[i] = 0;
        wt[i] = 0;
    }

    while (remaining > 0) {
        int min = INT_MAX;
        int idx = -1;

        // Find the process with the shortest remaining time that has
arrived
        for (int i = 0; i < n; i++) {
            if (at[i] <= time && !completed[i] && remainingTime[i] <
min) {
                min = remainingTime[i];
                idx = i;
            }
        }

        if (idx == -1) {
            time++;
            continue;
        }

        remainingTime[idx]--;
        if (remainingTime[idx] == 0) {
            completed[idx] = 1;
            remaining--;
            wt[idx] = time + 1 - at[idx] - bt[idx];
        }

        time++;
    }
}

void findTurnAroundTimePreemptive(int processes[], int n, int bt[], int
wt[], int tat[]) {
    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];
}

void printGanttChartPreemptive(int processes[], int n, int bt[], int
at[]) {
    int wt[n], tat[n];

    findWaitingTimePreemptive(processes, n, bt, wt, at);
    findTurnAroundTimePreemptive(processes, n, bt, wt, tat);

    // Print the Gantt chart
    printf("Gantt Chart (Preemptive):\n");

printf("--------------------------------------------------------\n");
```

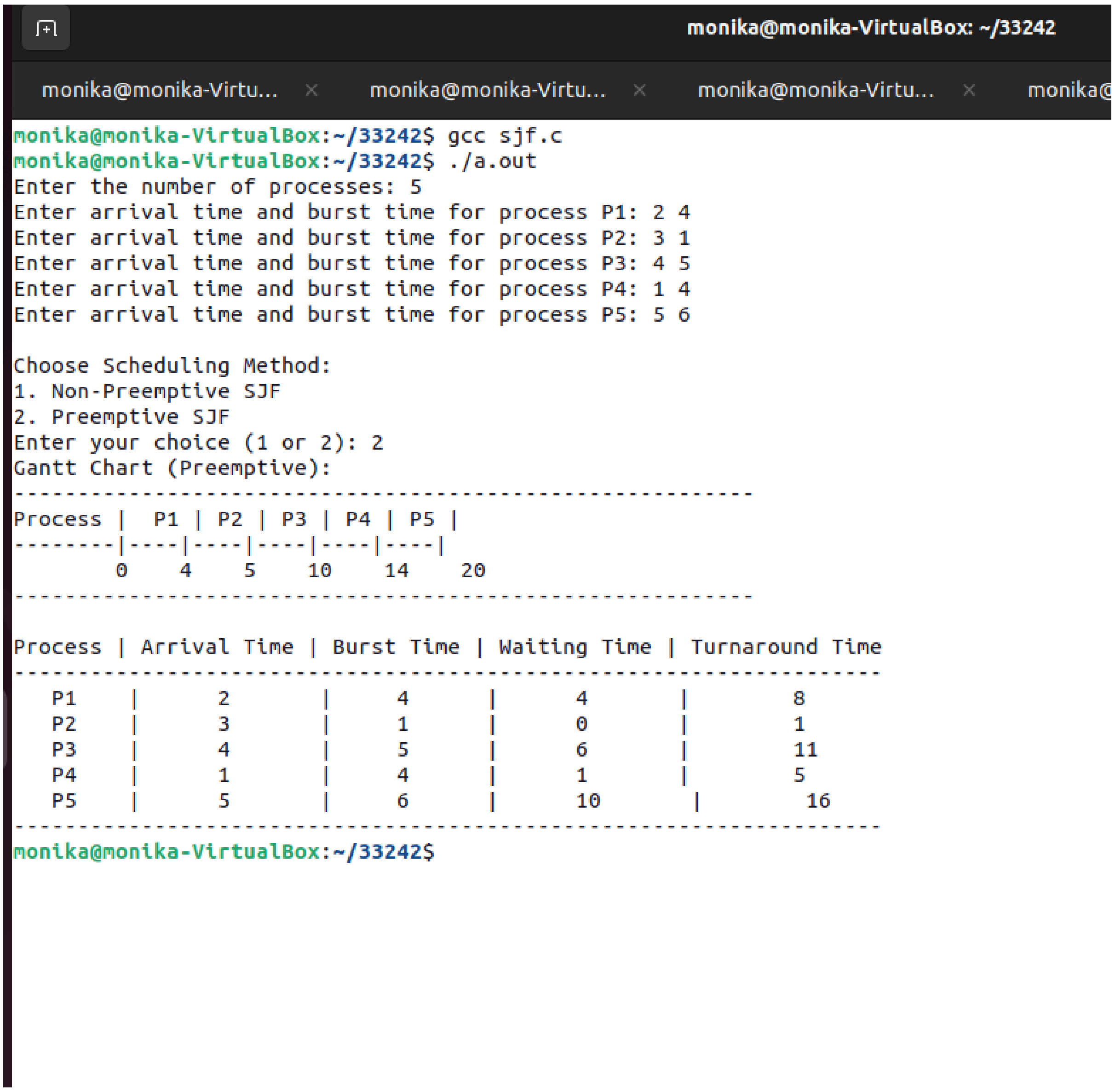
```
    printf("Process | ");

    for (int i = 0; i < n; i++)
        printf(" P%d |", processes[i]);
    printf("\n");
    printf("--------|");

    for (int i = 0; i < n; i++)
        printf("----|");
    printf("\n");

    int time = 0;
    printf("        %d", time);
    for (int i = 0; i < n; i++) {
        time += bt[i];
        printf("    %d", time);
    }
    printf("\n");

printf("----------------------------------------------------------\n");

    // Print table with results
    printf("\nProcess | Arrival Time | Burst Time | Waiting Time |
Turnaround Time\n");

printf("---------------------------------------------------------------
-----\n");
    for (int i = 0; i < n; i++) {
        printf("   P%d    |      %d       |     %d      |      %d
|        %d\n",
               processes[i], at[i], bt[i], wt[i], tat[i]);
    }

printf("---------------------------------------------------------------
-----\n");
}

int main() {
    int n, choice;

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

    int processes[n], bt[n], at[n];

    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;
        printf("Enter arrival time and burst time for process P%d: ",
processes[i]);
        scanf("%d %d", &at[i], &bt[i]);
    }
    printf("\nChoose Scheduling Method:\n");
    printf("1. Non-Preemptive SJF\n");
    printf("2. Preemptive SJF\n");
    printf("Enter your choice (1 or 2): ");
    scanf("%d", &choice);

    if (choice == 1) {
        printGanttChartNonPreemptive(processes, n, bt, at);
    } else if (choice == 2) {
        printGanttChartPreemptive(processes, n, bt, at);
    } else {
        printf("Invalid choice.\n");
    }

    return 0;
}
```

OUTPUT -



monika@monika-VirtualBox: ~/33242
monika@monika-VirtualBox: ~/...
monika@monika-VirtualBox: ~/...
monika@monika-VirtualB

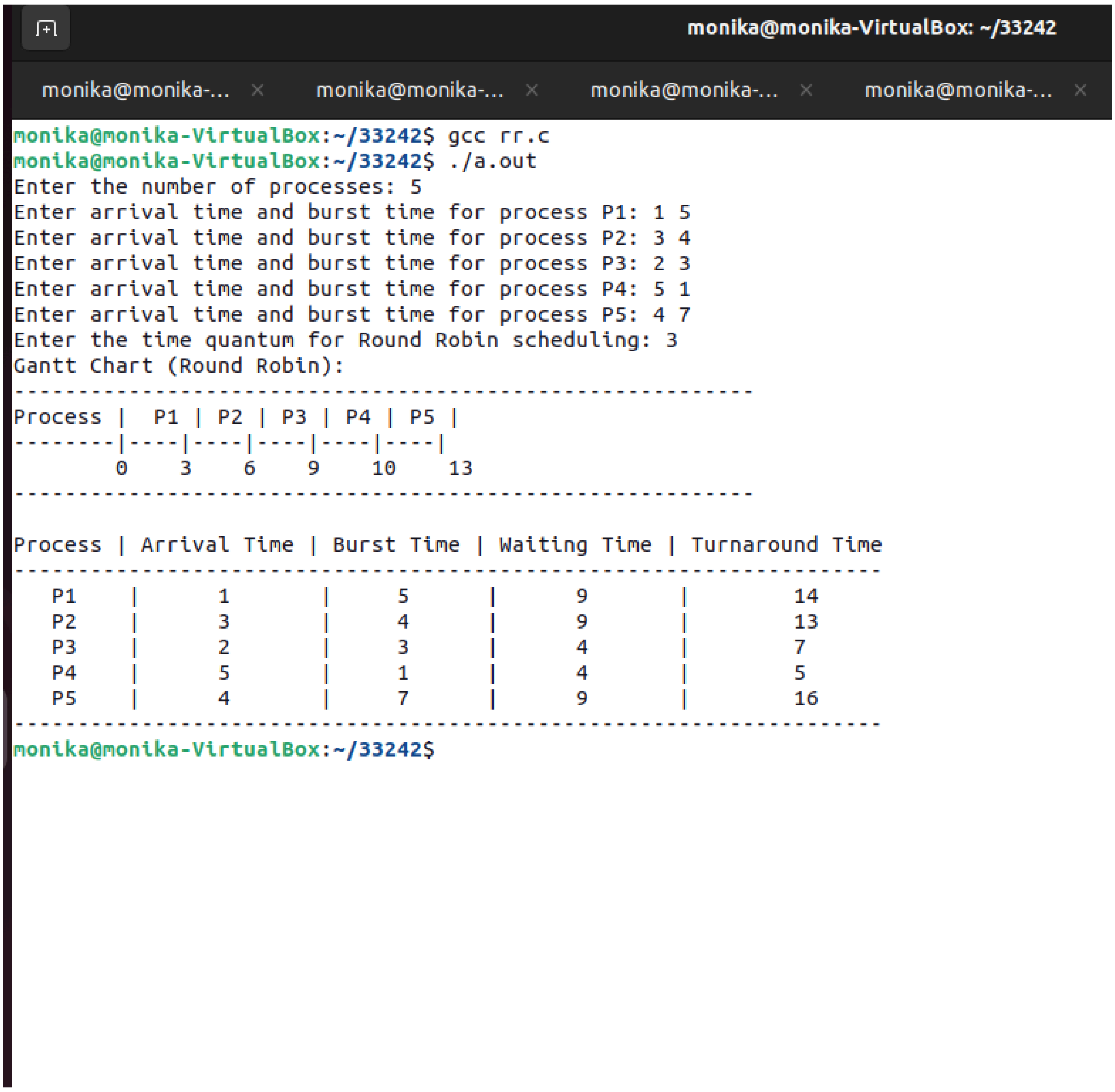
```
monika@monika-VirtualBox:~/33242$ gcc sjf.c
monika@monika-VirtualBox:~/33242$ ./a.out
Enter the number of processes: 5
Enter arrival time and burst time for process P1: 2 4
Enter arrival time and burst time for process P2: 3 5
Enter arrival time and burst time for process P3: 1 3
Enter arrival time and burst time for process P4: 5 2
Enter arrival time and burst time for process P5: 4 1

Choose Scheduling Method:
1. Non-Preemptive SJF
2. Preemptive SJF
Enter your choice (1 or 2): 1
Gantt Chart (Non-Preemptive):
----------------------------------------------------------
Process |  P1 | P2 | P3 | P4 | P5 |
--------|----|----|----|----|----|
        0    4    9    12    14    15
----------------------------------------------------------

Process | Arrival Time | Burst Time | Waiting Time | Turnaround Time
--------------------------------------------------------------------
   P1    |       2       |      4      |      5       |        9
   P2    |       3       |      5      |      8       |        13
   P3    |       1       |      3      |      0       |        3
   P4    |       5       |      2      |      0       |        2
   P5    |       4       |      1      |      0       |        1
--------------------------------------------------------------------
monika@monika-VirtualBox:~/33242$
```



```
monika@monika-VirtualBox:~/33242$ gcc sjf.c
monika@monika-VirtualBox:~/33242$ ./a.out
Enter the number of processes: 5
Enter arrival time and burst time for process P1: 2 4
Enter arrival time and burst time for process P2: 3 1
Enter arrival time and burst time for process P3: 4 5
Enter arrival time and burst time for process P4: 1 4
Enter arrival time and burst time for process P5: 5 6

Choose Scheduling Method:
1. Non-Preemptive SJF
2. Preemptive SJF
Enter your choice (1 or 2): 2
Gantt Chart (Preemptive):
----------------------------------------------------------
Process |  P1 | P2 | P3 | P4 | P5 |
--------|----|----|----|----|----|
        0    4    5    10    14    20
----------------------------------------------------------

Process | Arrival Time | Burst Time | Waiting Time | Turnaround Time
--------------------------------------------------------------------
   P1    |       2       |      4      |      4       |        8
   P2    |       3       |      1      |      0       |        1
   P3    |       4       |      5      |      6       |        11
   P4    |       1       |      4      |      1       |        5
   P5    |       5       |      6      |      10       |         16
--------------------------------------------------------------------
monika@monika-VirtualBox:~/33242$
```

Code for Round Robin

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

void findWaitingTimeRR(int processes[], int n, int bt[], int wt[], int
at[], int quantum) {
    int rem_bt[n];
    int t = 0; // Current time
    int completed = 0;

    // Initialize remaining burst times
    for (int i = 0; i < n; i++)
        rem_bt[i] = bt[i];

    while (completed < n) {
        int done = 0;

        for (int i = 0; i < n; i++) {
            if (rem_bt[i] > 0) {
                done = 1;

                if (rem_bt[i] > quantum) {
                    t += quantum;
                    rem_bt[i] -= quantum;
                } else {
                    t = t + rem_bt[i];
                    wt[i] = t - bt[i] - at[i];
                    rem_bt[i] = 0;
                    completed++;
                }
            }
        }

        if (done == 0)
            t++;
    }
}

void findTurnAroundTimeRR(int processes[], int n, int bt[], int wt[],
int tat[]) {
    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];
}

void printGanttChartRR(int processes[], int n, int bt[], int at[], int
quantum) {
    int wt[n], tat[n];
    int rem_bt[n];
    int t = 0; // Current time
    // Initialize remaining burst times
    for (int i = 0; i < n; i++)
        rem_bt[i] = bt[i];

    findWaitingTimeRR(processes, n, bt, wt, at, quantum);
    findTurnAroundTimeRR(processes, n, bt, wt, tat);

    // Print Gantt chart
    printf("Gantt Chart (Round Robin):\n");

printf("-----------------------------------------------------\n");
    printf("Process | ");

    for (int i = 0; i < n; i++)
        printf(" P%d |", processes[i]);
    printf("\n");
    printf("--------|");

    for (int i = 0; i < n; i++)
        printf("----|");
    printf("\n");

    t = 0;
    printf("        %d", t);
    for (int i = 0; i < n; i++) {
        t += (bt[i] > quantum) ? quantum : bt[i];
        printf("    %d", t);
    }
    printf("\n");

printf("-----------------------------------------------------\n");

    // Print table with results
    printf("\nProcess | Arrival Time | Burst Time | Waiting Time |
Turnaround Time\n");

printf("---------------------------------------------------------
-----\n");
    for (int i = 0; i < n; i++) {
        printf("   P%d    |      %d       |     %d      |      %d
|        %d\n",
               processes[i], at[i], bt[i], wt[i], tat[i]);
    }

printf("---------------------------------------------------------
-----\n");
}

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

    int processes[n], bt[n], at[n];

    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;
        printf("Enter arrival time and burst time for process P%d: ",
processes[i]);
        scanf("%d %d", &at[i], &bt[i]);
    }

    printf("Enter the time quantum for Round Robin scheduling: ");
    scanf("%d", &quantum);

    printGanttChartRR(processes, n, bt, at, quantum);

    return 0;
}
```
OUTPUT -



```
monika@monika-VirtualBox:~/33242$ gcc rr.c
monika@monika-VirtualBox:~/33242$ ./a.out
Enter the number of processes: 5
Enter arrival time and burst time for process P1: 1 5
Enter arrival time and burst time for process P2: 3 4
Enter arrival time and burst time for process P3: 2 3
Enter arrival time and burst time for process P4: 5 1
Enter arrival time and burst time for process P5: 4 7
Enter the time quantum for Round Robin scheduling: 3
Gantt Chart (Round Robin):
----------------------------------------------------------
Process |  P1 | P2 | P3 | P4 | P5 |
--------|----|----|----|----|----|
        0    3    6    9    10    13
----------------------------------------------------------

Process | Arrival Time | Burst Time | Waiting Time | Turnaround Time
--------------------------------------------------------------------
   P1    |       1       |     5      |      9       |        14
   P2    |       3       |     4      |      9       |        13
   P3    |       2       |     3      |      4       |        7
   P4    |       5       |     1      |      4       |        5
   P5    |       4       |     7      |      9       |        16
--------------------------------------------------------------------
monika@monika-VirtualBox:~/33242$
```

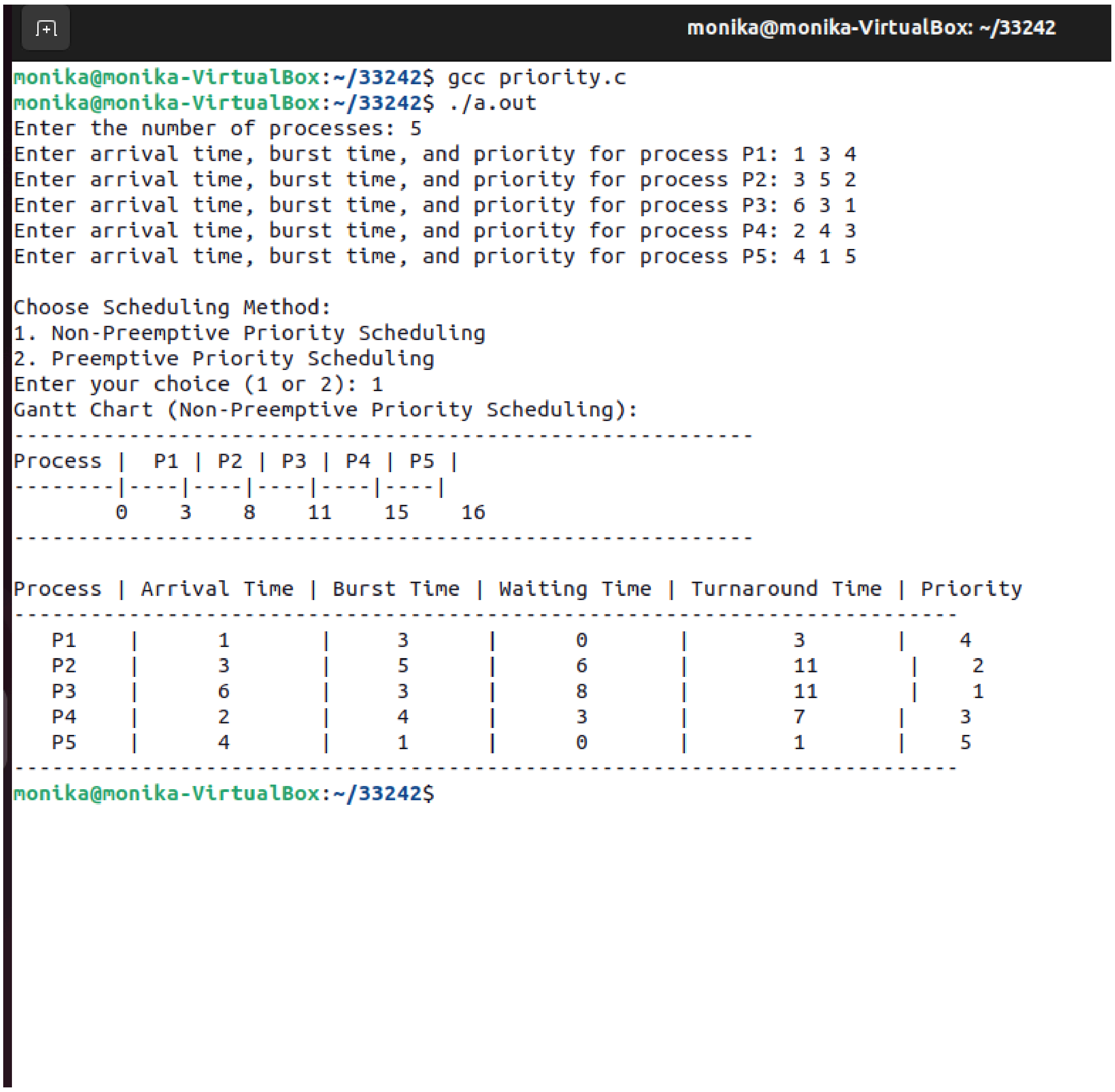
Code for Priority Scheduling Algorithm -

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

void findWaitingTimePriority(int processes[], int n, int bt[], int
wt[], int at[], int priority[], int preemptive) {
    int completed[n], remainingTime[n];
    int t = 0, completedCount = 0;

    for (int i = 0; i < n; i++) {
        remainingTime[i] = bt[i];
        completed[i] = 0;
        wt[i] = 0;
    }

    while (completedCount < n) {
        int maxPriority = -1, idx = -1;

        for (int i = 0; i < n; i++) {
            if (at[i] <= t && !completed[i] && (priority[i] >
maxPriority)) {
                maxPriority = priority[i];
                idx = i;
            }
        }

        if (idx != -1) {
            if (preemptive) {
                // Preemptive scheduling
                remainingTime[idx]--;
                if (remainingTime[idx] == 0) {
                    completed[idx] = 1;
                    completedCount++;
                    wt[idx] = t - bt[idx] - at[idx] + 1;
                }
            } else {
                // Non-preemptive scheduling
                t += remainingTime[idx];
                wt[idx] = t - bt[idx] - at[idx];
                completed[idx] = 1;
                completedCount++;
            }
        } else {
            t++;
        }

        if (preemptive) t++;
    }
}

void findTurnAroundTimePriority(int processes[], int n, int bt[], int
wt[], int tat[]) {
    for (int i = 0; i < n; i++)
        tat[i] = bt[i] + wt[i];
}

void printGanttChartPriority(int processes[], int n, int bt[], int
at[], int priority[], int preemptive) {
    int wt[n], tat[n];

    findWaitingTimePriority(processes, n, bt, wt, at, priority,
preemptive);
    findTurnAroundTimePriority(processes, n, bt, wt, tat);

    // Print Gantt chart
    printf("Gantt Chart (%s Priority Scheduling):\n", preemptive ?
"Preemptive" : "Non-Preemptive");
```

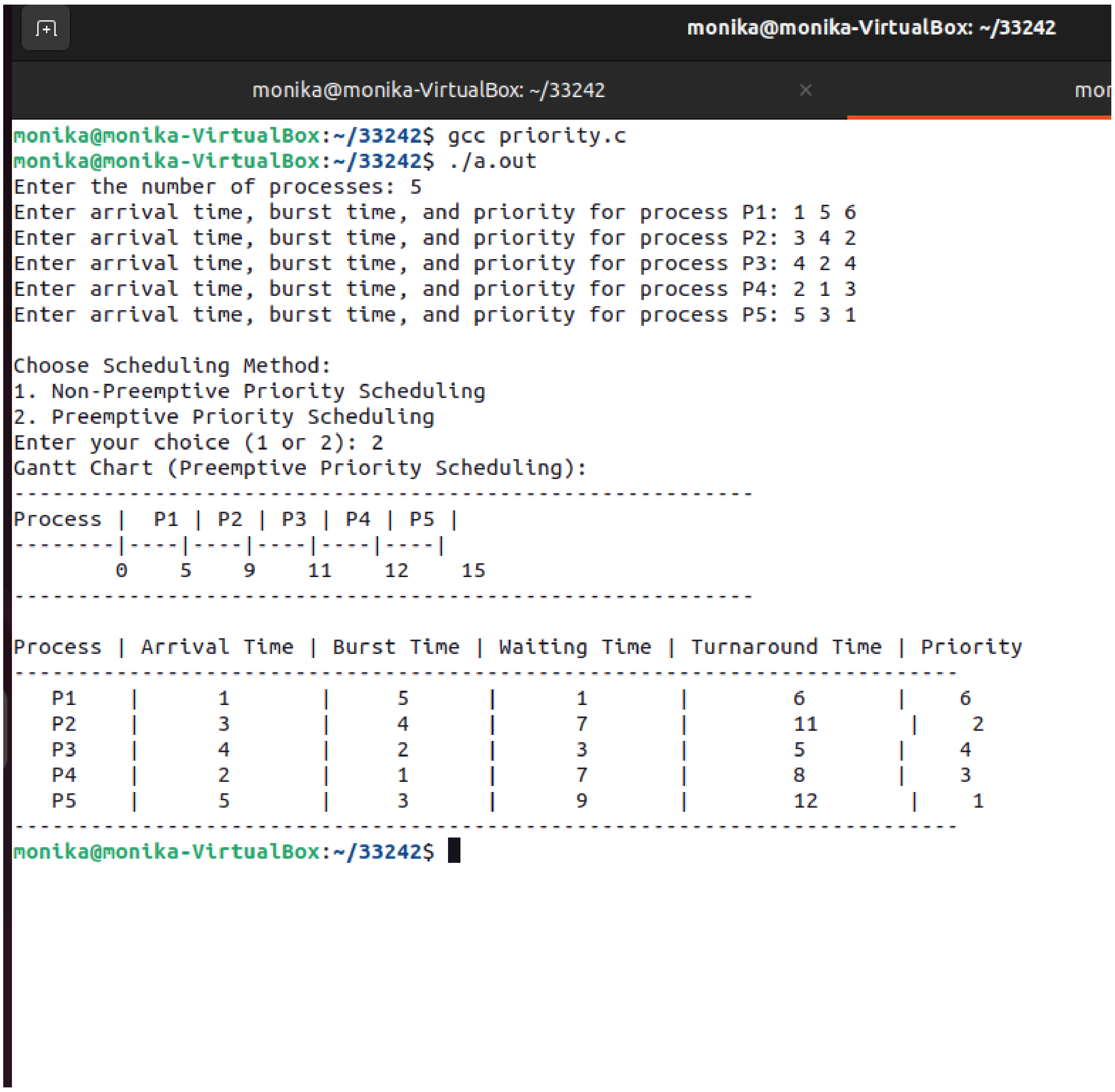
```
printf("------------------------------------------------------------\n");
    printf("Process | ");

    for (int i = 0; i < n; i++)
        printf(" P%d |", processes[i]);
    printf("\n");
    printf("--------|");

    for (int i = 0; i < n; i++)
        printf("----|");
    printf("\n");

    int t = 0;
    printf("        %d", t);
    for (int i = 0; i < n; i++) {
        t += bt[i];
        printf("    %d", t);
    }
    printf("\n");

printf("------------------------------------------------------------\n");

    // Print table with results
    printf("\nProcess | Arrival Time | Burst Time | Waiting Time |
Turnaround Time | Priority\n");

printf("----------------------------------------------------------------
-----------\n");
    for (int i = 0; i < n; i++) {
        printf("   P%d    |      %d       |     %d      |      %d
|        %d       |    %d\n",
               processes[i], at[i], bt[i], wt[i], tat[i], priority[i]);
    }

printf("----------------------------------------------------------------
-----------\n");
}

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

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

    int processes[n], bt[n], at[n], priority[n];

    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;
        printf("Enter arrival time, burst time, and priority for
process P%d: ", processes[i]);
```

```
        scanf("%d %d %d", &at[i], &bt[i], &priority[i]);
    }

    printf("\nChoose Scheduling Method:\n");
    printf("1. Non-Preemptive Priority Scheduling\n");
    printf("2. Preemptive Priority Scheduling\n");
    printf("Enter your choice (1 or 2): ");
    scanf("%d", &choice);

    preemptive = (choice == 2);

    printGanttChartPriority(processes, n, bt, at, priority, 
preemptive);

    return 0;
}
```

OUTPUT -



```
monika@monika-VirtualBox:~/33242$ gcc priority.c
monika@monika-VirtualBox:~/33242$ ./a.out
Enter the number of processes: 5
Enter arrival time, burst time, and priority for process P1: 1 3 4
Enter arrival time, burst time, and priority for process P2: 3 5 2
Enter arrival time, burst time, and priority for process P3: 6 3 1
Enter arrival time, burst time, and priority for process P4: 2 4 3
Enter arrival time, burst time, and priority for process P5: 4 1 5

Choose Scheduling Method:
1. Non-Preemptive Priority Scheduling
2. Preemptive Priority Scheduling
Enter your choice (1 or 2): 1
Gantt Chart (Non-Preemptive Priority Scheduling):
----------------------------------------------------------
Process |  P1 | P2 | P3 | P4 | P5 |
--------|----|----|----|----|----|
        0    3    8    11    15    16
----------------------------------------------------------

Process | Arrival Time | Burst Time | Waiting Time | Turnaround Time | Priority
--------------------------------------------------------------------------
   P1    |       1        |      3      |       0       |         3        |    4
   P2    |       3        |      5      |       6       |         11        |    2
   P3    |       6        |      3      |       8       |         11        |    1
   P4    |       2        |      4      |       3       |         7        |    3
   P5    |       4        |      1      |       0       |         1        |    5
--------------------------------------------------------------------------
monika@monika-VirtualBox:~/33242$
```


monika@monika-VirtualBox: ~/33242
monika@monika-VirtualBox: ~/33242

```
monika@monika-VirtualBox:~/33242$ gcc priority.c
monika@monika-VirtualBox:~/33242$ ./a.out
Enter the number of processes: 5
Enter arrival time, burst time, and priority for process P1: 1 5 6
Enter arrival time, burst time, and priority for process P2: 3 4 2
Enter arrival time, burst time, and priority for process P3: 4 2 4
Enter arrival time, burst time, and priority for process P4: 2 1 3
Enter arrival time, burst time, and priority for process P5: 5 3 1

Choose Scheduling Method:
1. Non-Preemptive Priority Scheduling
2. Preemptive Priority Scheduling
Enter your choice (1 or 2): 2
Gantt Chart (Preemptive Priority Scheduling):
----------------------------------------------------------
Process |  P1 | P2 | P3 | P4 | P5 |
--------|----|----|----|----|----|
        0    5    9    11    12    15
----------------------------------------------------------

Process | Arrival Time | Burst Time | Waiting Time | Turnaround Time | Priority
--------------------------------------------------------------------------
   P1    |       1       |      5      |      1       |        6        |     6
   P2    |       3       |      4      |      7       |        11        |     2
   P3    |       4       |      2      |      3       |        5        |     4
   P4    |       2       |      1      |      7       |        8        |     3
   P5    |       5       |      3      |      9       |        12        |     1
--------------------------------------------------------------------------
monika@monika-VirtualBox:~/33242$ 
```