

```

#include <stdio.h>

#include <limits.h> // For INT_MAX


struct Process {

    int pid;    // Process ID

    int arrival; // Arrival time

    int burst;  // Burst time

    int remaining; // Remaining burst time

    int waiting; // Waiting time

    int turnaround; // Turnaround time

    int completed; // Completion flag
};


int main() {

    int n, i, time = 0, completed = 0;

    float total_wait = 0, total_turnaround = 0;


    printf("Enter number of processes: ");

    scanf("%d", &n);


    struct Process p[n];


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

        p[i].pid = i + 1;

        printf("Enter Arrival Time of P%d: ", i + 1);

        scanf("%d", &p[i].arrival);

        printf("Enter Burst Time of P%d: ", i + 1);

        scanf("%d", &p[i].burst);

        p[i].remaining = p[i].burst;

        p[i].completed = 0;

    }
}

```

```
printf("\nGantt Chart:\n");
```

```
while (completed != n) {
```

```
    int idx = -1;
```

```
    int min_rem = INT_MAX;
```

```
    // Find process with minimum remaining time among arrived processes
```

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

```
        if (p[i].arrival <= time && p[i].completed == 0 && p[i].remaining < min_rem) {
```

```
            min_rem = p[i].remaining;
```

```
            idx = i;
```

```
        }
```

```
    }
```

```
    if (idx != -1) {
```

```
        printf(" | P%d (%d to %d) ", p[idx].pid, time, time + 1);
```

```
        p[idx].remaining--;
```

```
        time++;
```

```
        if (p[idx].remaining == 0) {
```

```
            p[idx].completed = 1;
```

```
            completed++;
```

```
            p[idx].turnaround = time - p[idx].arrival;
```

```
            p[idx].waiting = p[idx].turnaround - p[idx].burst;
```

```
            total_wait += p[idx].waiting;
```

```
            total_turnaround += p[idx].turnaround;
```

```
        }
```

```
    } else {
```

```
        // No process has arrived yet
```

```

        time++;
    }
}

printf("|\\n");

printf("\\n%-10s%-15s%-15s%-15s%-15s\\n", "Process", "Arrival", "Burst", "Waiting",
"Turnaround");

for (i = 0; i < n; i++) {
    printf("P%-9d%-15d%-15d%-15d%-15d\\n", p[i].pid, p[i].arrival, p[i].burst, p[i].waiting,
p[i].turnaround);
}

printf("\\nAverage Waiting Time: %.2f", total_wait / n);
printf("\\nAverage Turnaround Time: %.2f\\n", total_turnaround / n);

return 0;
}

```

#### OUTPUT

Enter number of processes: 4

Enter Arrival Time of P1: 0

Enter Burst Time of P1: 8

Enter Arrival Time of P2: 1

Enter Burst Time of P2: 4

Enter Arrival Time of P3: 2

Enter Burst Time of P3: 9

Enter Arrival Time of P4: 3

Enter Burst Time of P4: 5

Gantt Chart:

| P1 (0 to 1) | P2 (1 to 5) | P4 (5 to 10) | P1 (10 to 17) | P3 (17 to 26) |

| Process | Arrival | Burst | Waiting | Turnaround |
|---------|---------|-------|---------|------------|
| P1      | 0       | 8     | 9       | 17         |
| P2      | 1       | 4     | 0       | 4          |
| P3      | 2       | 9     | 15      | 24         |
| P4      | 3       | 5     | 2       | 7          |

Average Waiting Time: 6.50

Average Turnaround Time: 13.00