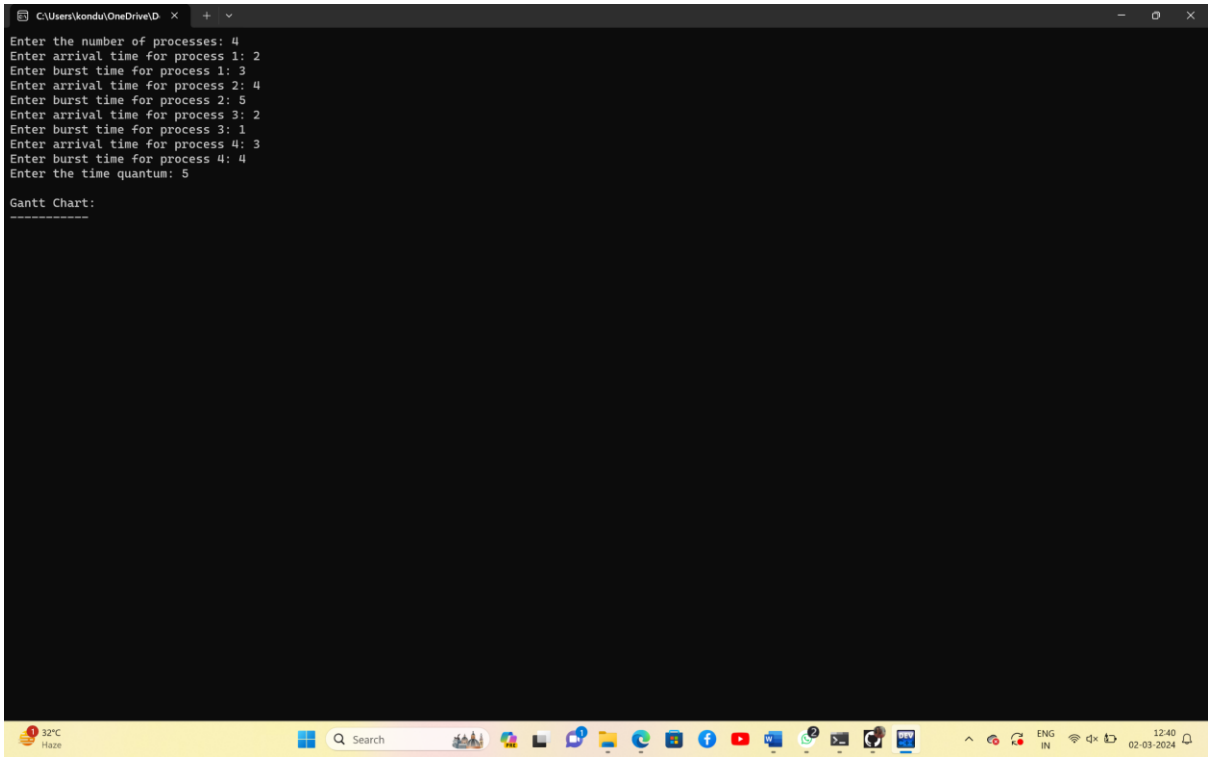
```
#include <stdio.h>

// Structure to represent a process
struct Process {
    int id;            // Process ID
    int arrivalTime;    // Arrival time
    int burstTime;      // Burst time
    int remainingTime;  // Remaining burst time
    int completionTime; // Completion time
};

// Function to perform Round Robin scheduling
void roundRobin(struct Process processes[], int n, int timeQuantum) {
    int currentTime = 0;
    int remainingProcesses = n;
    int queue[n]; // Queue to store process indices
    int front = 0, rear = -1;

    // Initialize remaining time for each process
    for (int i = 0; i < n; i++) {
        processes[i].remainingTime = processes[i].burstTime;
    }

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

    // Continue until all processes are completed
    while (remainingProcesses > 0) {
        // Enqueue processes that have arrived
        for (int i = 0; i < n; i++) {
            if (processes[i].arrivalTime <= currentTime && processes[i].remainingTime > 0) {
```

```
            queue[++rear] = i;
        }
    }

    // Dequeue and execute processes in the queue
    while (front <= rear) {
        int currentProcessIndex = queue[front++];
        struct Process *currentProcess = &processes[currentProcessIndex];
        printf("| P%d ", currentProcess->id);

        // Execute for time quantum or remaining time, whichever is smaller
        int executeTime = (currentProcess->remainingTime > timeQuantum) ? timeQuantum :
currentProcess->remainingTime;
        currentTime += executeTime;
        currentProcess->remainingTime -= executeTime;

        // Check if process is completed
        if (currentProcess->remainingTime == 0) {
            remainingProcesses--;
            currentProcess->completionTime = currentTime;
            printf("|");
        } else {
            // Enqueue the process again
            queue[++rear] = currentProcessIndex;
        }
    }
  }
}

int main() {
  int n; // Number of processes
```

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

struct Process processes[n]; // Array to store processes

// Input the process details
for (int i = 0; i < n; i++) {
    printf("Enter arrival time for process %d: ", i + 1);
    scanf("%d", &processes[i].arrivalTime);
    printf("Enter burst time for process %d: ", i + 1);
    scanf("%d", &processes[i].burstTime);
    processes[i].id = i + 1;
}

int timeQuantum;
printf("Enter the time quantum: ");
scanf("%d", &timeQuantum);

// Perform Round Robin scheduling
roundRobin(processes, n, timeQuantum);

printf("\n\n");

// Print completion time of each process
printf("Process Completion Times:\n");
printf("-------------------------\n");
printf("Process\tCompletion Time\n");
for (int i = 0; i < n; i++) {
    printf("P%d\t%d\n", processes[i].id, processes[i].completionTime);
}
```

```
    return 0;
}
```


C:\Users\kondu\OneDrive\D
Enter the number of processes: 4
Enter arrival time for process 1: 2
Enter burst time for process 1: 3
Enter arrival time for process 2: 4
Enter burst time for process 2: 5
Enter arrival time for process 3: 2
Enter burst time for process 3: 1
Enter arrival time for process 4: 3
Enter burst time for process 4: 4
Enter the time quantum: 5
Gantt Chart:

32°C
Haze
Search
ENG
IN
12:40
02-03-2024