```
    #include <stdio.h>
#define MAX_PROCESSES 100
int main() {
int n, quantum;
int burst_time[MAX_PROCESSES], remaining_time[MAX_PROCESSES];
int waiting_time[MAX_PROCESSES],
turnaround_time[MAX_PROCESSES];
int time = 0;
// Input number of processes
printf("Enter the number of processes: ");
scanf("%d", &n);
// Input burst time for each process
for (int i = 0; i < n; i++) {
printf("Enter burst time for Process %d: ", i + 1);
scanf("%d", &burst_time[i]);
remaining_time[i] = burst_time[i]; // Initialize remaining time
waiting_time[i] = 0;
}
// Input time quantum
printf("Enter the time quantum: ");
scanf("%d", &quantum);
// Round Robin Scheduling Simulation
int done;
do {
done = 1;
for (int i = 0; i < n; i++) {
if (remaining_time[i] > 0) {
done = 0; // There is still a pending process
if (remaining_time[i] > quantum) {
time += quantum;
remaining_time[i] -= quantum;
} else {
time += remaining_time[i];
waiting_time[i] = time - burst_time[i];
remaining_time[i] = 0;
```

```
}
}
}
} while (!done);
// Calculate turnaround time
for (int i = 0; i < n; i++) {
turnaround_time[i] = burst_time[i] + waiting_time[i];
}
// Display results
printf("\n------------------------------------------\n");
printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");
printf("------------------------------------------\n");
for (int i = 0; i < n; i++) {
printf("P%d\t%d\t\t%d\t\t%d\n", i + 1, burst_time[i],
waiting_time[i], turnaround_time[i]);
}
printf("------------------------------------------\n");
return 0;
}
```