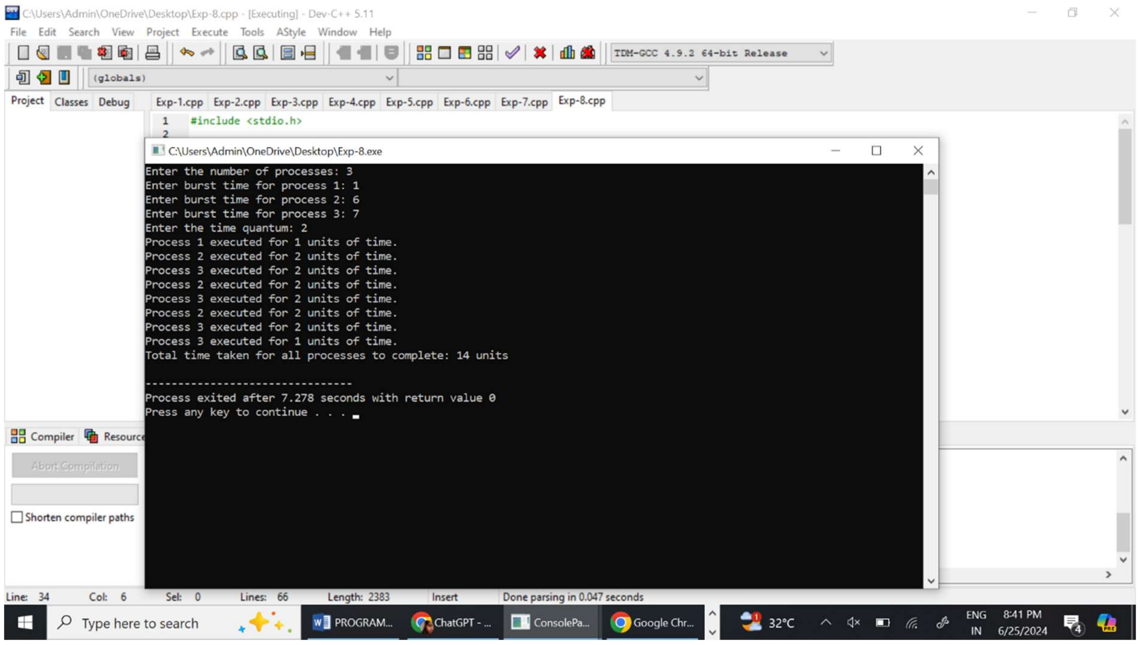
Program For Optimizing Task Management with Round Robin Scheduling

```
#include <stdio.h>


// Structure to represent a process

typedef struct {

   int pid;    // Process ID

   int burst_time;  // Burst Time of the process

   int remaining_time; // Remaining Time of the process

} Process;


// Function to perform Round Robin scheduling

void roundRobinScheduling(Process processes[], int n, int time_quantum) {

   int total_time = 0;  // Total time passed

   int complete = 0;    // Number of processes completed


   // Loop until all processes are completed

   while (complete < n) {

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

         // If process has remaining time greater than 0

         if (processes[i].remaining_time > 0) {

            // Check if remaining time is less than or equal to time quantum

            if (processes[i].remaining_time <= time_quantum) {

               total_time += processes[i].remaining_time;

               printf("Process %d executed for %d units of time.\n", processes[i].pid,
processes[i].remaining_time);

               processes[i].remaining_time = 0;  // Process is completed

               complete++;  // Increment the number of completed processes

            } else {

               // Process runs for the time quantum

               total_time += time_quantum;
```

```
                processes[i].remaining_time -= time_quantum;
                printf("Process %d executed for %d units of time.\n", processes[i].pid, time_quantum);
            }
        }
    }
}

printf("Total time taken for all processes to complete: %d units\n", total_time);
}

int main() {
    int n;  // Number of processes
    int time_quantum;  // Time Quantum

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

    Process processes[n];

    // Get the burst time of each process
    for (int i = 0; i < n; i++) {
        processes[i].pid = i + 1;
        printf("Enter burst time for process %d: ", i + 1);
        scanf("%d", &processes[i].burst_time);
        processes[i].remaining_time = processes[i].burst_time;
    }

    // Get the time quantum
    printf("Enter the time quantum: ");
    scanf("%d", &time_quantum);
    // Perform Round Robin Scheduling
    roundRobinScheduling(processes, n, time_quantum);

    return 0;
}
```

OUTPUT



```
Enter the number of processes: 3
Enter burst time for process 1: 1
Enter burst time for process 2: 6
Enter burst time for process 3: 7
Enter the time quantum: 2
Process 1 executed for 1 units of time.
Process 2 executed for 2 units of time.
Process 3 executed for 2 units of time.
Process 2 executed for 2 units of time.
Process 3 executed for 2 units of time.
Process 2 executed for 2 units of time.
Process 3 executed for 2 units of time.
Process 3 executed for 1 units of time.
Total time taken for all processes to complete: 14 units

--------------------------------
Process exited after 7.278 seconds with return value 0
Press any key to continue . . .
```