

6. Construct a C program to implement pre-emptive priority scheduling algorithm.

```
#include <stdio.h>
#include <limits.h>
int main() {
    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    int at[n], bt[n], pr[n], rem[n];
    for (int i = 0; i < n; i++) {
        printf("Enter arrival time of P%d: ", i + 1);
        scanf("%d", &at[i]);
        printf("Enter burst time of P%d: ", i + 1);
        scanf("%d", &bt[i]);
        printf("Enter priority of P%d (lower number = higher priority): ", i + 1);
        scanf("%d", &pr[i]);
        rem[i] = bt[i]; // remaining burst time
    }
    int time = 0, completed = 0;
    int wt[n], tat[n];
    for (int i = 0; i < n; i++) { wt[i] = 0; tat[i] = 0; }
    printf("\nGantt Chart:\n");
    while (completed < n) {
        int idx = -1;
        int highestPr = INT_MAX;
        for (int i = 0; i < n; i++) {
            if (at[i] <= time && rem[i] > 0) {
                if (pr[i] < highestPr) {
                    highestPr = pr[i];
                    idx = i;
                }
            }
        }
        if (idx == -1) {
            printf("| idle ");
            time++;
            continue;
        }
        printf("| P%d ", idx + 1);
        rem[idx]--;
        time++;
        if (rem[idx] == 0) {
            completed++;
            tat[idx] = time - at[idx];
            wt[idx] = tat[idx] - bt[idx];
        }
    }
}
```

```

}
printf("\n");
printf("\nProcess\tArrival\tBurst\tPriority\tWaiting\tTurnaround\n");
double totalWT = 0, totalTAT = 0;
for (int i = 0; i < n; i++) {
    printf("P%d\t%d\t%d\t%d\t%d\t%d\t%d\n", i + 1, at[i], bt[i], pr[i], wt[i], tat[i]);
    totalWT += wt[i];
    totalTAT += tat[i];
}
printf("\nAverage Waiting Time = %.2f", totalWT / n);
printf("\nAverage Turnaround Time = %.2f\n", totalTAT / n);
return 0;
}

```

INPUT:

Enter number of processes: 3
 Enter arrival time of P1: 0
 Enter burst time of P1: 10
 Enter priority of P1 (lower number = higher priority): 2
 Enter arrival time of P2: 1
 Enter burst time of P2: 5
 Enter priority of P2 (lower number = higher priority): 1
 Enter arrival time of P3: 2
 Enter burst time of P3: 8
 Enter priority of P3 (lower number = higher priority): 3

OUTPUT:

The screenshot shows the execution of the C++ program 'os lab 1.cpp' in a debugger. The code implements a preemptive Round Robin scheduler. It finds the highest priority process, then iterates through processes, scheduling them until completion or a time limit. The Gantt chart shows the timeline for four processes (P1-P4) over 14 time units. The output window displays the user input and the generated Gantt chart and resource utilization table.

Gantt Chart:

```

| idle | idle | P4 | P4 | P4 | P4 | P4 | P2 | P2 | P2 | P2 | P2 |
| P1 | P2 | P2 | P1 | P3 |
| P3 |

```

Resource Utilization Table:

Process	Arrival	Burst	Priority	Waiting	Turnaround
P1	7	8	8	6	14
P2	5	7	3	1	8
P3	6	7	8	15	22
P4	2	4	0	0	4

Average Waiting Time = 5.50
Average Turnaround Time = 12.00