NAME: G. GANESH

REG NO: 192373008

EXERICSE-6

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

Aim:

To construct a C program to implement the pre-emptive priority scheduling algorithm, where processes are scheduled based on their priority, and pre-emption occurs if a higher-priority process arrives during execution.

Algorithm:

1. Input the number of processes, their burst times, priorities, and arrival times.
2. Initialize a ready queue and keep track of the remaining burst time for each process.
3. At every time unit:
    - Check if a higher-priority process has arrived.
    - Preempt the currently running process if necessary.
4. Calculate waiting time and turnaround time for each process.
5. Compute the average waiting and turnaround times.
6. Display the scheduling order, waiting times, turnaround times, and averages.

Procedure:

1. Read the number of processes, burst times, priorities, and arrival times.
2. Implement the preemptive priority scheduling algorithm by simulating time unit by time unit.
3. Calculate the waiting and turnaround times for each process.
4. Compute the average waiting and turnaround times.
5. Display the results.

Code:

```
#include <stdio.h>

#include <limits.h>

int main() {

   int n, i, completed = 0, current_time = 0, shortest = -1;

   printf("Enter the number of processes: ");

   scanf("%d", &n);

   int burst_time[n], remaining_time[n], priority[n], arrival_time[n];

   int waiting_time[n], turnaround_time[n], is_completed[n];
```

```
    float avg_waiting_time = 0, avg_turnaround_time = 0;
    printf("Enter the arrival time, burst time, and priority of the processes:\n");
    for (i = 0; i < n; i++) {
        printf("Process %d - Arrival Time: ", i + 1);
        scanf("%d", &arrival_time[i]);
        printf("Process %d - Burst Time: ", i + 1);
        scanf("%d", &burst_time[i]);
        printf("Process %d - Priority: ", i + 1);
        scanf("%d", &priority[i]);
        remaining_time[i] = burst_time[i];
        is_completed[i] = 0; // Mark all processes as not completed
    }
    while (completed < n) {
        int highest_priority = INT_MAX;
        for (i = 0; i < n; i++) {
            if (!is_completed[i] && arrival_time[i] <= current_time && priority[i] < highest_priority) {
                highest_priority = priority[i];
                shortest = i;
            }
        }
        if (shortest == -1) {
            current_time++;
            continue;
        }
        remaining_time[shortest]--;
        if (remaining_time[shortest] == 0) {
            completed++;
            is_completed[shortest] = 1;
            int finish_time = current_time + 1;
            turnaround_time[shortest] = finish_time - arrival_time[shortest];
```

```
    waiting_time[shortest] = turnaround_time[shortest] - burst_time[shortest];
        }
        current_time++;
    }
    for (i = 0; i < n; i++) {
        avg_waiting_time += waiting_time[i];
        avg_turnaround_time += turnaround_time[i];
    }
    avg_waiting_time /= n;
    avg_turnaround_time /= n;
    printf("\nProcess\tArrival Time\tBurst Time\tPriority\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", i + 1, arrival_time[i], burst_time[i],
priority[i], waiting_time[i], turnaround_time[i]);
    }
    printf("\nAverage Waiting Time: %.2f\n", avg_waiting_time);
    printf("Average Turnaround Time: %.2f\n", avg_turnaround_time);
    return 0;
}
```

Result:

The preemptive priority scheduling program successfully executes the process with the highest priority at each time unit, calculates waiting and turnaround times, and computes averages. The program displays the process execution details and timing information.

Output:

```
Enter the number of processes: 3
Enter the arrival time, burst time, and priority of the processes:
Process 1 - Arrival Time: 0
Process 1 - Burst Time: 7
Process 1 - Priority: 2
Process 2 - Arrival Time: 2
Process 2 - Burst Time: 4
Process 2 - Priority: 1
Process 3 - Arrival Time: 4
Process 3 - Burst Time: 1
Process 3 - Priority: 3

Process Arrival Time    Burst Time      Priority        Waiting Time     Turnaround Time
1       0               7               2               4       11
2       2               4               1               0       4
3       4               1               3               7       8

Average Waiting Time: 3.67
Average Turnaround Time: 7.67


...Program finished with exit code 0
Press ENTER to exit console.
```