Name: P.MAL REDDY

Reg-No: 192372024

5. Construct a scheduling program with C that selects the waiting process with the highest priority to execute next.

Aim:

To develop a scheduling program in C that selects and executes the waiting process with the highest priority using priority scheduling.

Algorithm:

1. **Input Process Details**: Collect process IDs, burst times, and priorities.
2. **Sort by Priority**: Arrange processes in descending order of priority. For equal priorities, sort by arrival time.
3. **Execute Processes**: Execute processes sequentially based on priority and calculate waiting time (WT) and turnaround time (TAT).
4. **Output Results**: Display process order, waiting times, turnaround times, and average times.

Procedure:

1. Define a structure to represent processes (process ID, burst time, priority).
2. Accept input for the number of processes and their details.
3. Sort processes based on priority.
4. Calculate WT and TAT for each process.
5. Execute the processes in sorted order and display the output.

Code:

```
#include <stdio.h>

typedef struct {

    int pid, burst_time, priority, waiting_time, turnaround_time;

} Process;

void sortByPriority(Process p[], int n) {

    for (int i = 0; i < n - 1; i++) {
```

```
        for (int j = 0; j < n - i - 1; j++) {
            if (p[j].priority < p[j + 1].priority) {
                Process temp = p[j];
                p[j] = p[j + 1];
                p[j + 1] = temp;
            }
        }
    }
}
void calculateTimes(Process p[], int n) {
    p[0].waiting_time = 0;
    for (int i = 1; i < n; i++)
        p[i].waiting_time = p[i - 1].waiting_time + p[i - 1].burst_time;
    for (int i = 0; i < n; i++)
        p[i].turnaround_time = p[i].waiting_time + p[i].burst_time;
}
void displayResults(Process p[], int n) {
    printf("PID\tPriority\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++)
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\n", p[i].pid, p[i].priority, p[i].burst_time,
p[i].waiting_time, p[i].turnaround_time);
}
int main() {
    int n;
```

```
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    Process p[n];
    for (int i = 0; i < n; i++) {
        printf("Enter PID, Burst Time, and Priority for Process %d: ", i + 1);
        scanf("%d %d %d", &p[i].pid, &p[i].burst_time, &p[i].priority);
    }
    sortByPriority(p, n);
    calculateTimes(p, n);
    displayResults(p, n);
    return 0;
}
```

Result:

- The program accepts process details and displays a schedule based on the highest priority.

Output: