EXPERIMENT-5

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

AIM:-

To design a CPU scheduling program in C that implements Priority Scheduling, where the process with the highest priority is selected to execute next. Lower numbers indicate higher priority (e.g., priority 1 is higher than priority 2).

ALGORITHM:-

- Input Process Details:

Read the number of processes.

Input the burst times and priorities for each process.

- Sort Processes by Priority:

Arrange processes in ascending order of priority.

If two processes have the same priority, order them based on their arrival time (if applicable).

- Calculate Completion Time (CT):

For the first process, CT = Burst Time.

For subsequent processes, CT = CT(previous) + Burst Time.

- Calculate Turnaround Time (TAT):

TAT = CT - Arrival Time (Assume arrival time is 0).

- Calculate Waiting Time (WT):

WT = TAT - Burst Time.

- Calculate Average TAT and WT:

Compute the average turnaround time and waiting time.

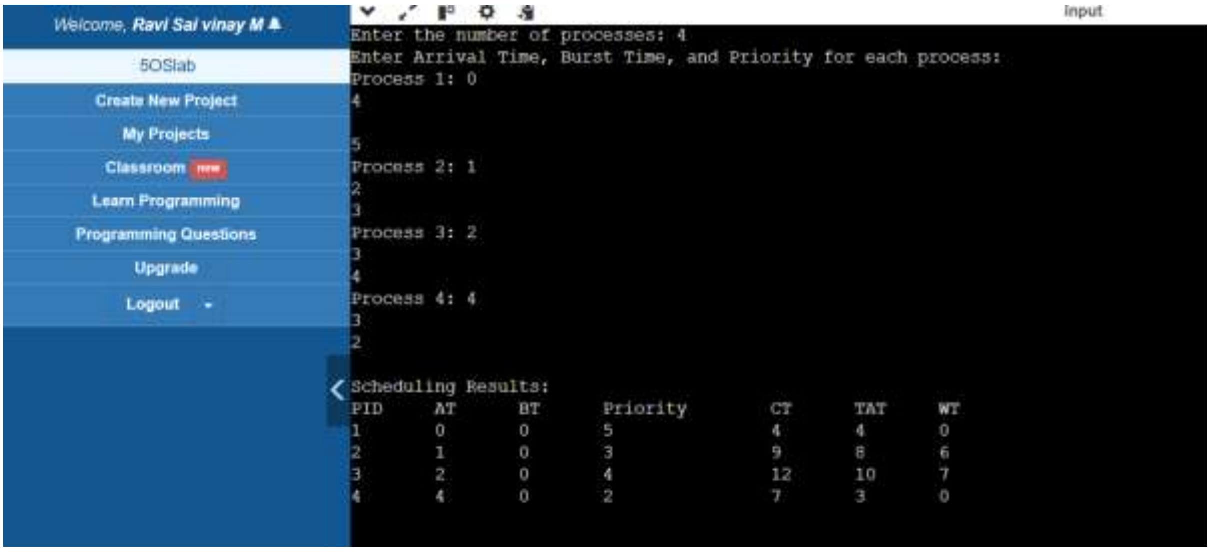
- Display Results:

Show process IDs, burst times, priorities, completion times, turnaround times, and waiting times, along with averages.

CODE:-

```
#include <stdio.h>


typedef struct {
    int process_id;
    int burst_time;
    int priority;
    int completion_time;
    int turnaround_time;
    int waiting_time;
} Process;


void sort_by_priority(Process processes[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (processes[j].priority > processes[j + 1].priority) {
                // Swap processes[j] and processes[j + 1]
                Process temp = processes[j];
                processes[j] = processes[j + 1];
                processes[j + 1] = temp;
            }
        }
    }
}

int main() {
    int n;
    float avg_turnaround_time = 0, avg_waiting_time = 0;

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

    Process processes[n];

    // Input burst time and priority for each process
    printf("Enter the burst time and priority for each process:\n");
    for (int i = 0; i < n; i++) {
        processes[i].process_id = i + 1;
        printf("Process %d - Burst Time: ", i + 1);
        scanf("%d", &processes[i].burst_time);
        printf("Process %d - Priority: ", i + 1);
        scanf("%d", &processes[i].priority);
    }

    // Sort processes by priority
    sort_by_priority(processes, n);

    // Calculate Completion Time (CT), Turnaround Time (TAT), and Waiting Time (WT)
    processes[0].completion_time = processes[0].burst_time;
    processes[0].turnaround_time = processes[0].completion_time; // Since Arrival Time = 0
    processes[0].waiting_time = processes[0].turnaround_time - processes[0].burst_time;

    for (int i = 1; i < n; i++) {
        processes[i].completion_time = processes[i - 1].completion_time + processes[i].burst_time;
        processes[i].turnaround_time = processes[i].completion_time; // Since Arrival Time = 0
        processes[i].waiting_time = processes[i].turnaround_time - processes[i].burst_time;
    }

    // Calculate averages
    for (int i = 0; i < n; i++) {
        avg_turnaround_time += processes[i].turnaround_time;
        avg_waiting_time += processes[i].waiting_time;
    }
    avg_turnaround_time /= n;
    avg_waiting_time /= n;

    // Display results
    printf("\nProcess\tBurst Time\tPriority\tCompletion Time\tTurnaround Time\tWaiting Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n", processes[i].process_id, processes[i].burst_time,
               processes[i].priority, processes[i].completion_time, processes[i].turnaround_time, processes[i].waiting_time);
    }

    printf("\nAverage Turnaround Time: %.2f\n", avg_turnaround_time);
    printf("Average Waiting Time: %.2f\n", avg_waiting_time);

    return 0;
}
```

OUTPUT:-



```
Enter the number of processes: 4
Enter Arrival Time, Burst Time, and Priority for each process:
Process 1: 0
4

5
Process 2: 1
2
3
Process 3: 2
3
4
Process 4: 4
3
2

Scheduling Results:
PID     AT      BT      Priority        CT      TAT     WT
1       0       0       5               4       4       0
2       1       0       3               9       8       6
3       2       0       4               12      10      7
4       4       0       2               7       3       0
```

RESULT:-

The Priority Scheduling program was successfully implemented in C. The process with the highest priority (lowest priority number) was executed first, and the program calculated the completion, turnaround, and waiting times for all processes, along with their average