**EXPERIMENT NO. 02**

|  |
| --- |
| NAME: Ayush Vinod Upadhyay  ROLL NO: I025  SAP ID: 60003220131  BRANCH: Information Technology  BATCH: 1 |

#include <stdio.h>

void calculateWaitingTime(int processes[], int n, int burst\_time[], int waiting\_time[])

{

    waiting\_time[0] = 0; // Waiting time for the first process is 0

    for (int i = 1; i < n; i++)

    {

        waiting\_time[i] = burst\_time[i - 1] + waiting\_time[i - 1];

    }

}

void calculateTurnaroundTime(int processes[], int n, int burst\_time[], int waiting\_time[], int turnaround\_time[])

{

    for (int i = 0; i < n; i++)

    {

        turnaround\_time[i] = burst\_time[i] + waiting\_time[i];

    }

}

void displayGanttChart(int processes[], int n, int burst\_time[])

{

    printf("\nGantt Chart:\n");

    for (int i = 0; i < n; i++)

    {

        printf("| P%d ", processes[i]);

    }

    printf("|\n");

    int current\_time = 0;

    for (int i = 0; i < n; i++)

    {

        printf("%d\t", current\_time);

        current\_time += burst\_time[i];

    }

    printf("%d\n", current\_time);

}

void calculateAverageWaitingTime(int processes[], int n, int burst\_time[], int waiting\_time[])

{

    float total\_waiting\_time = 0;

    for (int i = 0; i < n; i++)

    {

        total\_waiting\_time += waiting\_time[i];

    }

    float avg\_waiting\_time = total\_waiting\_time / n;

    printf("Average Waiting Time: %.2f\n", avg\_waiting\_time);

}

int main()

{

    int n;

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

    scanf("%d", &n);

    int processes[n], burst\_time[n], waiting\_time[n], turnaround\_time[n];

    for (int i = 0; i < n; i++)

    {

        printf("Enter burst time for process P%d: ", i + 1);

        scanf("%d", &burst\_time[i]);

        processes[i] = i + 1;

    }

    calculateWaitingTime(processes, n, burst\_time, waiting\_time);

    calculateTurnaroundTime(processes, n, burst\_time, waiting\_time, turnaround\_time);

    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");

    for (int i = 0; i < n; i++)

    {

        printf("%d\t%d\t\t%d\t\t%d\n", processes[i], burst\_time[i], waiting\_time[i], turnaround\_time[i]);

    }

    displayGanttChart(processes, n, burst\_time);

    calculateAverageWaitingTime(processes, n, burst\_time, waiting\_time);

    return 0;

}

**Q2**

#include <stdio.h>

struct Process

{

    int id;

    int arrival\_time;

    int burst\_time;

    int priority;

    int waiting\_time;

    int turnaround\_time;

};

void sjf\_with\_priority(struct Process processes[], int n)

{

    int total\_waiting\_time = 0;

    int total\_turnaround\_time = 0;

    for (int i = 0; i < n; i++)

    {

        for (int j = i + 1; j < n; j++)

        {

            if (processes[i].arrival\_time > processes[j].arrival\_time || (processes[i].arrival\_time == processes[j].arrival\_time &&

                                                                          (processes[i].priority > processes[j].priority ||

                                                                           (processes[i].priority == processes[j].priority && processes[i].burst\_time > processes[j].burst\_time))))

            {

                struct Process temp = processes[i];

                processes[i] = processes[j];

                processes[j] = temp;

            }

        }

    }

    int current\_time = 0;

    for (int i = 0; i < n; i++)

    {

        if (processes[i].arrival\_time > current\_time)

        {

            current\_time = processes[i].arrival\_time;

        }

        processes[i].waiting\_time = current\_time - processes[i].arrival\_time;

        processes[i].turnaround\_time = processes[i].waiting\_time + processes[i].burst\_time;

        total\_waiting\_time += processes[i].waiting\_time;

        total\_turnaround\_time += processes[i].turnaround\_time;

        current\_time += processes[i].burst\_time;

    }

    printf("\nGantt Chart:\n");

    printf("0");

    for (int i = 0; i < n; i++)

    {

        printf("->P%d->%d", processes[i].id, current\_time);

    }

    printf("\n\nTABLE\n");

    printf("Process AT BT WT TAT\n");

    for (int i = 0; i < n; i++)

    {

        printf("P%d\t%d\t%d\t%d\t%d\n", processes[i].id, processes[i].arrival\_time, processes[i].burst\_time, processes[i].waiting\_time, processes[i].turnaround\_time);

    }

    double avg\_waiting\_time = (double)total\_waiting\_time / n;

    double avg\_turnaround\_time = (double)total\_turnaround\_time / n;

    printf("\nAverage Turnaround Time: %.6lf\n", avg\_turnaround\_time);

    printf("Average Waiting Time: %.6lf\n", avg\_waiting\_time);

}

int main()

{

    int n;

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

    scanf("%d", &n);

    struct Process processes[n];

    for (int i = 0; i < n; i++)

    {

        processes[i].id = i + 1;

        printf("Enter the arrival time for process P%d: ", i + 1);

        scanf("%d", &processes[i].arrival\_time);

        printf("Enter the burst time for process P%d: ", i + 1);

        scanf("%d", &processes[i].burst\_time);

        printf("Enter the priority for process P%d: ", i + 1);

        scanf("%d", &processes[i].priority);

    }

    sjf\_with\_priority(processes, n);

    return 0;

}

**Q3**

#include <stdio.h>

void priorityScheduling(int processes[], int n, int burst\_time[], int priority[], int arrival\_time[])

{

    int waiting\_time[n], turnaround\_time[n];

    for (int i = 0; i < n - 1; i++)

    {

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

        {

            if (arrival\_time[j] > arrival\_time[j + 1])

            {

                int temp = arrival\_time[j];

                arrival\_time[j] = arrival\_time[j + 1];

                arrival\_time[j + 1] = temp;

                temp = priority[j];

                priority[j] = priority[j + 1];

                priority[j + 1] = temp;

                temp = burst\_time[j];

                burst\_time[j] = burst\_time[j + 1];

                burst\_time[j + 1] = temp;

                temp = processes[j];

                processes[j] = processes[j + 1];

                processes[j + 1] = temp;

            }

        }

    }

    waiting\_time[0] = 0;

    int current\_time = arrival\_time[0];

    for (int i = 1; i < n; i++)

    {

        waiting\_time[i] = burst\_time[i - 1] + waiting\_time[i - 1];

        current\_time += burst\_time[i - 1];

    }

    for (int i = 0; i < n; i++)

    {

        turnaround\_time[i] = burst\_time[i] + waiting\_time[i];

    }

    printf("\nGantt Chart:\n");

    for (int i = 0; i < n; i++)

    {

        printf("| P%d ", processes[i]);

    }

    printf("|\n");

    current\_time = arrival\_time[0];

    for (int i = 0; i < n; i++)

    {

        printf("%d\t", current\_time);

        current\_time += burst\_time[i];

    }

    printf("%d\n", current\_time);

    printf("\nProcess\tArrival Time\tBurst Time\tPriority\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\t\t%d\n", processes[i], arrival\_time[i], burst\_time[i], priority[i], waiting\_time[i], turnaround\_time[i]);

    }

    float avg\_waiting\_time = 0, avg\_turnaround\_time = 0;

    for (int 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("\nAverage Waiting Time: %.2f\n", avg\_waiting\_time);

    printf("Average Turnaround Time: %.2f\n", avg\_turnaround\_time);

}

int main()

{

    int n;

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

    scanf("%d", &n);

    int processes[n], burst\_time[n], priority[n], arrival\_time[n];

    for (int i = 0; i < n; i++)

    {

        printf("Enter arrival time for process P%d: ", i + 1);

        scanf("%d", &arrival\_time[i]);

        printf("Enter burst time for process P%d: ", i + 1);

        scanf("%d", &burst\_time[i]);

        printf("Enter priority for process P%d: ", i + 1);

        scanf("%d", &priority[i]);

        processes[i] = i + 1;

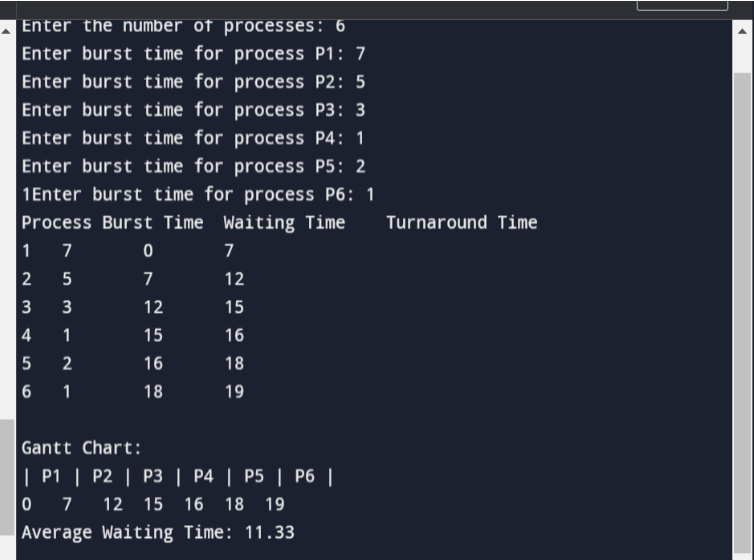
    }

    priorityScheduling(processes, n, burst\_time, priority, arrival\_time);

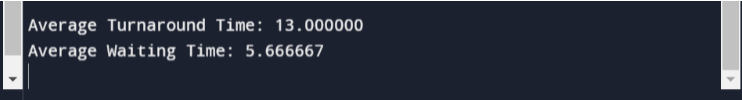
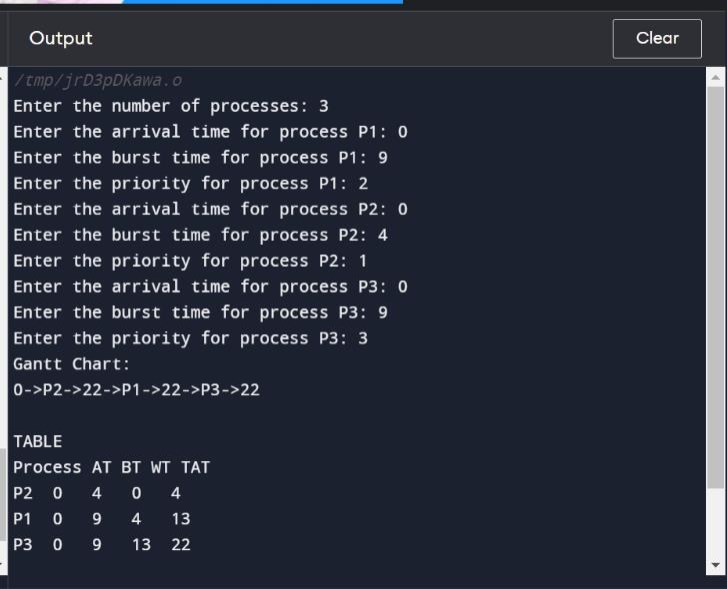
    return 0;

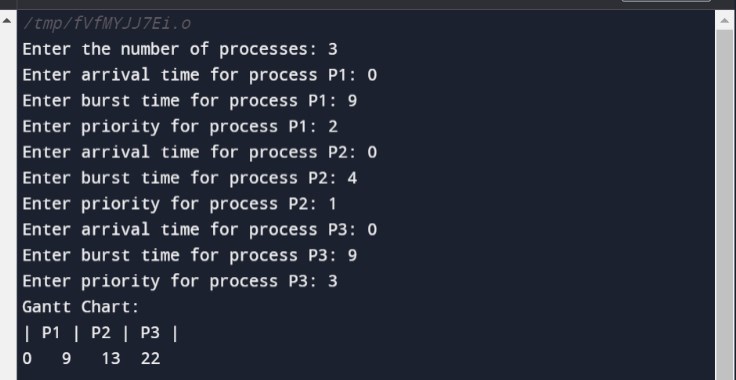
}

**OUTPUT:**

Q1

**Q2**



**Q3**

