Academic Year: 2023-24 Sem: III
Sub: Operating Systems Laboratory SAP ID: 60003220045
Name: Anish Sharma

EXPERIMENT NO. 02

Code:
Q1

```
#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;
```

A.Y.: 2023-24

}

OUTPUT:

Q1

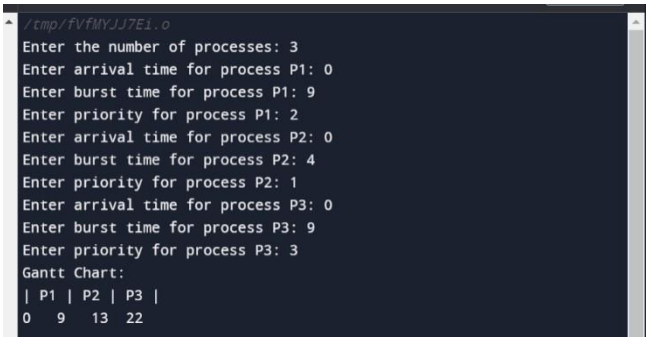
```
Enter the number of processes: 6
Enter burst time for process P1: 7
Enter burst time for process P2: 5
Enter burst time for process P3: 3
Enter burst time for process P4: 1
Enter burst time for process P5: 2
1Enter burst time for process P6: 1
Process Burst Time  Waiting Time    Turnaround Time
1   7       0       7
2   5       7       12
3   3       12      15
4   1       15      16
5   2       16      18
6   1       18      19

Gantt Chart:
| P1 | P2 | P3 | P4 | P5 | P6 |
0   7   12  15  16  18  19
Average Waiting Time: 11.33
```

Q2

```
Output                                                    Clear
/tmp/jrD3pDKawa.o
Enter the number of processes: 3
Enter the arrival time for process P1: 0
Enter the burst time for process P1: 9
Enter the priority for process P1: 2
Enter the arrival time for process P2: 0
Enter the burst time for process P2: 4
Enter the priority for process P2: 1
Enter the arrival time for process P3: 0
Enter the burst time for process P3: 9
Enter the priority for process P3: 3
Gantt Chart:
0->P2->22->P1->22->P3->22

TABLE
Process AT BT WT TAT
P2  0   4   0   4
P1  0   9   4   13
P3  0   9   13  22

Average Turnaround Time: 13.000000
Average Waiting Time: 5.666667
```

```
Process Arrival Time    Burst Time  Priority    Waiting Time
    Turnaround Time
1   0       9       2       0       9
2   0       4       1       9       13
3   0       9       3       13      22

Average Waiting Time: 7.33
Average Turnaround Time: 14.67
```

Q3.


/tmp/fVfMYJJ7Ei.o
Enter the number of processes: 3
Enter arrival time for process P1: 0
Enter burst time for process P1: 9
Enter priority for process P1: 2
Enter arrival time for process P2: 0
Enter burst time for process P2: 4
Enter priority for process P2: 1
Enter arrival time for process P3: 0
Enter burst time for process P3: 9
Enter priority for process P3: 3
Gantt Chart:
| P1 | P2 | P3 |
0 9 13 22