CSB 302: Operating System

Lab 3: Pre-emptive Job Scheduling

Submitted By:

Name: **Umang Kumar**

Roll No: **201210051**

Branch: **CSE**

Semester: **5th Sem**

Submitted To: ***Dr. Rishav Singh***

NATIONAL INSTITUTE OF TECHNOLOGY DELHI

Department of Computer Science and Engineering

2022

Q1) Write a program in C to implement Pre-emptive Shortest Job First CPU scheduling algorithm.

CODE:

```
#include <stdio.h>
using namespace std;

int main() {
   int arrival_time[10], burst_time[10], temp[10];
   int i, smallest, count = 0, time, limit;
   double wait_time = 0, turnaround_time = 0, end;
   float average_waiting_time, average_turnaround_time;

   printf("Enter the Total Number of Processes:");
   scanf("%d", &limit);
   printf("Enter Details of %d Processes\n", limit);

   for (i = 0; i < limit; i++) {
       printf("\nEnter Arrival Time:");
       scanf("%d", &arrival_time[i]);
       printf("Enter Burst Time:");
       scanf("%d", &burst_time[i]);
       temp[i] = burst_time[i];
   }

   burst_time[9] = 9999;
   for (time = 0; count != limit; time++) {
       smallest = 9;
       for (i = 0; i < limit; i++) {
           if (arrival_time[i] <= time &&
               burst_time[i] < burst_time[smallest] && burst_time[i] > 0) {
               smallest = i;
           }
       }

       burst_time[smallest]--;
       if (burst_time[smallest] == 0) {
           count++;
           end = time + 1;
           wait_time =
               wait_time + end - arrival_time[smallest] - temp[smallest];
           turnaround_time = turnaround_time + end - arrival_time[smallest];
```

```
        }
    }

    average_waiting_time = wait_time / limit;
    average_turnaround_time = turnaround_time / limit;

    printf("\nAverage Waiting Time:%lf\n", average_waiting_time);
    printf("Average Turnaround Time:%lf\n", average_turnaround_time);
    return 0;
}
```

Output:

```
Enter the Total Number of Processes:4
Enter Details of 4 Processes

Enter Arrival Time:1
Enter Burst Time:4

Enter Arrival Time:2
Enter Burst Time:4

Enter Arrival Time:3
Enter Burst Time:5

Enter Arrival Time:4
Enter Burst Time:8

Average Waiting Time:4.750000
Average Turnaround Time:10.000000
```

Q2) Write a program in C to implement the Round Robin CPU scheduling algorithm.

CODE:

```
#include <stdio.h>

int main() {
    int i, limit, total = 0, x, counter = 0, time_quantum;
    int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_time[10],
        temp[10];
    float average_wait_time, average_turnaround_time;
```

```
printf("\nEnter Total Number of Processes:\t");
scanf("%d", &limit);
x = limit;

for (i = 0; i < limit; i++) {
    printf("\nEnter Details of Process[%d]\n", i + 1);
    printf("Arrival Time:\t");
    scanf("%d", &arrival_time[i]);
    printf("Burst Time:\t");
    scanf("%d", &burst_time[i]);
    temp[i] = burst_time[i];
}

printf("\nEnter Time Quantum:\t");
scanf("%d", &time_quantum);
printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting Time\n");

for (total = 0, i = 0; x != 0;) {
    if (temp[i] <= time_quantum && temp[i] > 0) {
        total = total + temp[i];
        temp[i] = 0;
        counter = 1;
    } else if (temp[i] > 0) {
        temp[i] = temp[i] - time_quantum;
        total = total + time_quantum;
    }

    if (temp[i] == 0 && counter == 1) {
        x--;
        printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1, burst_time[i],
               total - arrival_time[i],
               total - arrival_time[i] - burst_time[i]);

        wait_time = wait_time + total - arrival_time[i] - burst_time[i];
        turnaround_time = turnaround_time + total - arrival_time[i];
        counter = 0;
    }

    if (i == limit - 1) {
        i = 0;
    } else if (arrival_time[i + 1] <= total) {
```

```
            i++;
        } else {
            i = 0;
        }
    }

    average_wait_time = wait_time * 1.0 / limit;
    average_turnaround_time = turnaround_time * 1.0 / limit;

    printf("\n\nAverage Waiting Time:\t%f", average_wait_time);
    printf("\nAvg Turnaround Time:\t%f\n", average_turnaround_time);
    return 0;
}
```

Output:

```
Enter Total Number of Processes:        4

Enter Details of Process[1]
Arrival Time:   0
Burst Time:     4

Enter Details of Process[2]
Arrival Time:   1
Burst Time:     7

Enter Details of Process[3]
Arrival Time:   2
Burst Time:     5

Enter Details of Process[4]
Arrival Time:   3
Burst Time:     6

Enter Time Quantum:     3

Process ID              Burst Time      Turnaround Time         Waiting Time

Process[1]              4               13                      9
Process[3]              5               16                      11
Process[4]              6               18                      12
Process[2]              7               21                      14
```

```
Average Waiting Time:    11.500000
Avg Turnaround Time:     17.000000
```