Lab Assignment-11

ROLL: 2005535 | NAME: SAHIL SINGH | DATE: 05/04/22

QUES 1: WAP in C to implement the Round Robin scheduling algorithm.

SOLUTION:

```c
#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: ");
    scanf("%d", &limit);
    x = limit;
    for (i = 0; i < limit; i++)
    {
        printf("\nEnter details for P[%d]\n", i + 1);
        printf("Arrival Time: ");
        scanf("%d", &arrival_time[i]);
        printf("Burst Time: ");
        scanf("%d", &burst_time[i]);
        temp[i] = burst_time[i];
        printf("\n");
    }
    printf("Enter Time Quantum: ");
    scanf("%d", &time_quantum);
    printf("\nProcess\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("\nP[%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: %.2f", average_wait_time);
    printf("\nAvg Turnaround Time: %.2f\n", average_turnaround_time);
    return 0;
}
```

OUTPUT:

```
Enter Total Number of Processes: 4

Enter details for P[1]
Arrival Time: 0
Burst Time: 8


Enter details for P[2]
Arrival Time: 1
Burst Time: 5


Enter details for P[3]
Arrival Time: 2
Burst Time: 10


Enter details for P[4]
Arrival Time: 3
Burst Time: 11

Enter Time Quantum: 6

Process                 Burst Time       Turnaround Time         Waiting Time
P[2]            5                10                      5
P[1]            8                25                      17
P[3]            10               27                      17
```

```
P[4]            11               31                       20

Average Waiting Time: 14.75
Avg Turnaround Time: 23.25
```

QUES 2: WAP in C to implement the Priority scheduling algorithm.

SOLUTION:

```c
#include <stdio.h>
int main()
{
    int a[10], b[10], x[10], pr[10];
    int waiting[10], turnaround[10], completion[10];
    int i, j, smallest, count = 0, time, n;
    double avg = 0, tt = 0, end;
    printf("\nEnter the number of Processes: ");
    scanf("%d", &n);
    printf("\n");
    for (i = 0; i < n; i++)
    {
        printf("Enter arrival time of P[%d]: ", i + 1);
        scanf("%d", &a[i]);
    }
    printf("\n");
    for (i = 0; i < n; i++)
    {
        printf("Enter burst time of P[%d]: ", i + 1);
        scanf("%d", &b[i]);
    }
    printf("\n");
    for (i = 0; i < n; i++)
    {
        printf("Enter priority of P[%d]: ", i + 1);
        scanf("%d", &pr[i]);
    }
    for (i = 0; i < n; i++)
        x[i] = b[i];
    pr[9] = 100000;
    for (time = 0; count != n; time++)
    {
        smallest = 9;
        for (i = 0; i < n; i++)
        {
            if (a[i] <= time && pr[smallest] > pr[i] && b[i] > 0)
                smallest = i;
        }
        b[smallest] = b[smallest] - 1;
        if (b[smallest] == 0)
        {
            count++;
            waiting[smallest] = time + 1 - a[smallest] - x[smallest];
            turnaround[smallest] = time + 1 - a[smallest];
            end = time + 1;
            completion[smallest] = end;
        }
    }
    printf("Process\t   Burst-time\t   Arrival-time\t   Waiting-time\t   Turnaround-
time\t   Completion-time\t   Priority\n");
    for (i = 0; i < n; i++)
    {
        printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t\t%d\t\t\t%d\n", i + 1, x[i], a[i],
waiting[i], turnaround[i], completion[i], pr[i]);
        avg = avg + waiting[i];
        tt = tt + turnaround[i];
    }
    printf("\n\nAverage waiting time: %.3f", (avg / n));
    printf("\nAverage Turnaround time: %.3f\n", (tt / n));
}
```

OUTPUT:

```
Enter the number of Processes: 5

Enter arrival time of P[1]: 0
Enter arrival time of P[2]: 1
Enter arrival time of P[3]: 3
Enter arrival time of P[4]: 2
Enter arrival time of P[5]: 4

Enter burst time of P[1]: 3
Enter burst time of P[2]: 6
Enter burst time of P[3]: 1
Enter burst time of P[4]: 2
Enter burst time of P[5]: 4

Enter priority of P[1]: 3
Enter priority of P[2]: 4
Enter priority of P[3]: 9
Enter priority of P[4]: 7
Enter priority of P[5]: 8
Process    Burst-time     Arrival-time    Waiting-time    Turnaround-time        Completion-time        Priority
P1              3               0               0               3                       3                       3
P2              6               1               2               8                       9                       4
P3              1               3               12              13                      16                      9
P4              2               2               7               9                       11                      7
P5              4               4               7               11                      15                      8
Average waiting time: 5.600
Average Turnaround time: 8.800
```
