Week 08

8. Implement Non-pre-emptive/Pre-emptive CPU Scheduling Algorithms to Find Turnaround Time and Waiting Time (Minimum 2 from All Process Scheduling Algorithms)

A) FIRST COME FIRST SERVE (FCFS):

AIM: To write a C program to simulate the CPU scheduling algorithm **First Come First Serve (FCFS)**.

DESCRIPTION:

The average waiting time using the FCFS algorithm is calculated by first keeping the waiting time of the first process as zero. The waiting time for the second process is the burst time of the first process, and the waiting time for the third process is the sum of the burst times of the first and second processes, and so on. After calculating all the waiting times, the average waiting time is calculated as the average of all the waiting times. The FCFS algorithm serves the process that arrives first.

ALGORITHM:

1. Start the process
2. Accept the number of processes in the ready queue
3. For each process in the ready queue, assign the process name and burst time
4. Set the waiting time of the first process as '0' and its burst time as its turnaround time
5. For each process in the ready queue, calculate:
    - Waiting time (n) = Waiting time (n-1) + Burst time (n-1)
    - Turnaround time (n) = Waiting time (n) + Burst time (n)
6. Calculate:
    - Average waiting time = Total waiting time / Number of processes

- Average turnaround time = Total turnaround time / Number of processes

7. Stop the process

SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>
main() {
   int bt[20], wt[20], tat[20], i, n;
   float wtavg, tatavg;
   system("clear");
   printf("\nEnter the number of processes -- ");
   scanf("%d", &n);
   for(i = 0; i < n; i++) {
      printf("\nEnter Burst Time for Process %d -- ", i);
      scanf("%d", &bt[i]);
   }
   wt[0] = wtavg = 0;
   tat[0] = tatavg = bt[0];
   for(i = 1; i < n; i++) {
      wt[i] = wt[i-1] + bt[i-1];
      tat[i] = tat[i-1] + bt[i];
      wtavg = wtavg + wt[i];
      tatavg = tatavg + tat[i];
   }
   printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME
\n");
   for(i = 0; i < n; i++)
      printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
   printf("\nAverage Waiting Time -- %f", wtavg / n);
   printf("\nAverage Turnaround Time -- %f\n", tatavg / n);
   getchar();
}
```

INPUT:

```
Enter the number of processes -- 3
Enter Burst Time for Process 0 -- 24
Enter Burst Time for Process 1 -- 3
Enter Burst Time for Process 2 -- 3
```

OUTPUT:

```
PROCESS BURST TIME WAITING TIME TURNAROUND TIME
P0 24 0 24
P1 3 24 27
P2 3 27 30
Average Waiting Time -- 17.000000
Average Turnaround Time -- 27.000000
```

B) SHORTEST JOB FIRST (Non-Preemption):

AIM: To write a program to simulate the CPU scheduling algorithm **Shortest Job First (SJF) (Non-preemption)**.

DESCRIPTION:

To calculate the average waiting time in the Shortest Job First algorithm, the processes are sorted by their burst time in ascending order. Then, the waiting time of each process is calculated by summing up the burst times of all processes that arrive before that process.

ALGORITHM:

1. Start the process
2. Accept the number of processes in the ready queue
3. For each process in the ready queue, assign the process ID and accept the CPU burst time
4. Sort the ready queue by burst time in ascending order
5. Set the waiting time of the first process as '0' and its turnaround time as its burst time

6. For each process in the ready queue, calculate:
    - Waiting time(n) = Waiting time(n-1) + Burst time(n-1)
    - Turnaround time(n) = Waiting time(n) + Burst time(n)
7. Calculate:
    - Average waiting time = Total waiting time / Number of processes
    - Average turnaround time = Total turnaround time / Number of processes
8. Stop the process

SOURCE CODE:

```
#include <stdio.h>
#include <conio.h>
void main() {
   int p[20], bt[20], wt[20], tat[20], i, k, n, temp;
   float wtavg, tatavg;
   system("clear");
   printf("\nEnter the number of processes -- ");
   scanf("%d", &n);
   for(i = 0; i < n; i++) {
      p[i] = i;
      printf("Enter Burst Time for Process %d -- ", i);
      scanf("%d", &bt[i]);
   }
   for(i = 0; i < n; i++) {
      for(k = i + 1; k < n; k++) {
         if(bt[i] > bt[k]) {
            temp = bt[i];
            bt[i] = bt[k];
            bt[k] = temp;
            temp = p[i];
            p[i] = p[k];
            p[k] = temp;
         }
      }
```

```
    }
    wt[0] = wtavg = 0;
    tat[0] = tatavg = bt[0];
    for(i = 1; i < n; i++) {
        wt[i] = wt[i-1] + bt[i-1];
        tat[i] = tat[i-1] + bt[i];
        wtavg = wtavg + wt[i];
        tatavg = tatavg + tat[i];
    }
    printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME
\n");
    for(i = 0; i < n; i++)
        printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
    printf("\nAverage Waiting Time -- %f", wtavg / n);
    printf("\nAverage Turnaround Time -- %f", tatavg / n);
    getchar();
}
```

INPUT:

```
Enter the number of processes -- 4
Enter Burst Time for Process 0 -- 6
Enter Burst Time for Process 1 -- 8
Enter Burst Time for Process 2 -- 7
Enter Burst Time for Process 3 -- 3
```

OUTPUT:

```
PROCESS BURST TIME WAITING TIME TURNAROUND TIME
P3 3 0 3
P0 6 3 9
P2 7 9 16
P1 8 16 24
Average Waiting Time -- 7.000000
Average Turnaround Time -- 13.000000
```

C) ROUND ROBIN:

AIM: To simulate the CPU scheduling algorithm **Round Robin**.

DESCRIPTION:

The goal is to calculate the average waiting time. A time slice is used where each process gets executed within that time-slice. If the process doesn't finish in one time slice, it goes back to the queue. We check if the burst time is less than or equal to the time slice and calculate the turnaround time and waiting time accordingly.

ALGORITHM:

1. Start the process
2. Accept the number of processes in the ready queue and the time quantum (time slice)
3. For each process in the ready queue, assign the process ID and accept the CPU burst time
4. Calculate the number of time slices for each process:
    - Time slices for process(n) = burst time(n) / time slice
5. If burst time is less than the time slice, then number of time slices = 1.
6. Treat the ready queue as a circular queue:
    - Waiting time for process(n) = waiting time of process(n-1) + burst time of process(n-1) + the time difference in getting the CPU
    - Turnaround time for process(n) = waiting time of process(n) + burst time of process(n) + the time difference in getting the CPU
7. Calculate:
    - Average waiting time = Total waiting time / Number of processes
    - Average turnaround time = Total turnaround time / Number of processes
8. Stop the process

SOURCE CODE:

```
#include <stdio.h>
#include<stdlib.h>

void main() {
   int i, j, n, bu[10], wa[10], tat[10], t, ct[10], max;
   float awt = 0, att = 0, temp = 0;
   system("clear");
   printf("Enter the no of processes -- ");
   scanf("%d", &n);
   for(i = 0; i < n; i++) {
      printf("\nEnter Burst Time for process %d -- ", i+1);
      scanf("%d", &bu[i]);
      ct[i] = bu[i];
   }
   printf("\nEnter the size of time slice -- ");
   scanf("%d", &t);
   max = bu[0];
   for(i = 1; i < n; i++) {
      if(max < bu[i]) max = bu[i];
   }

   for(j = 0; j < (max/t) + 1; j++) {
      for(i = 0; i < n; i++) {
         if(bu[i] != 0) {
            if(bu[i] <= t) {
               tat[i] = temp + bu[i];
               temp = temp + bu[i];
               bu[i] = 0;
            } else {
               bu[i] = bu[i] - t;
               temp = temp + t;
            }
         }
      }
   }
```

```
    for(i = 0; i < n; i++) {
        wa[i] = tat[i] - ct[i];
        att += tat[i];
        awt += wa[i];
    }
    printf("\nThe Average Turnaround time is -- %f", att / n);
    printf("\nThe Average Waiting time is -- %f", awt / n);
    printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND TIME
\n");
    for(i = 0; i < n; i++)
        printf("\t%d \t %d \t\t %d \t\t %d \n", i+1, ct[i], wa[i], tat[i]);
    getchar();
}
```

INPUT:

```
Enter the no of processes – 3
Enter Burst Time for process 1 – 24
Enter Burst Time for process 2 -- 3
Enter Burst Time for process 3 – 3
Enter the size of time slice – 3
```

OUTPUT:

```
PROCESS BURST TIME WAITING TIME TURNAROUND TIME
1 24 6 30
2 3 4 7
3 3 7 10
The Average Turnaround time is – 15.666667
The Average Waiting time is – 5.666667
```

D) PRIORITY:

AIM: To write a C program to simulate the CPU scheduling **Priority algorithm**.

DESCRIPTION:

The priority algorithm calculates the average waiting time by sorting the processes based on their priorities. The waiting time for each process is obtained by summing up the burst times of all the previous processes.

ALGORITHM:

1. Start the process
2. Accept the number of processes in the ready queue
3. For each process in the ready queue, assign the process ID and accept the CPU burst time
4. Sort the ready queue according to priority numbers.
5. Set the waiting time of the first process as '0' and its burst time as its turnaround time
6. Arrange the processes based on their priority
7. For each process in the ready queue, calculate:
    - Waiting time(n) = Waiting time(n-1) + Burst time(n-1)
    - Turnaround time(n) = Waiting time(n) + Burst time(n)
8. Calculate:
    - Average waiting time = Total waiting time / Number of processes
    - Average turnaround time = Total turnaround time / Number of processes
9. Stop the process

SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>

void main() {
   int p[20], bt[20], pri[20], wt[20], tat[20], i, k, n, temp;
   float wtavg, tatavg;
   system("clear");
```

```
    printf("Enter the number of processes --- ");
    scanf("%d", &n);
    for(i = 0; i < n; i++) {
        p[i] = i;
        printf("Enter the Burst Time & Priority of Process %d --- ", i);
        scanf("%d %d", &bt[i], &pri[i]);
    }
    for(i = 0; i < n; i++) {
        for(k = i + 1; k < n; k++) {
            if(pri[i] > pri[k]) {
                temp = p[i];
                p[i] = p[k];
                p[k] = temp;
                temp = bt[i];
                bt[i] = bt[k];
                bt[k] = temp;
                temp = pri[i];
                pri[i] = pri[k];
                pri[k] = temp;
            }
        }
    }
    wtavg = wt[0] = 0;
    tatavg = tat[0] = bt[0];
    for(i = 1; i < n; i++) {
        wt[i] = wt[i-1] + bt[i-1];
        tat[i] = tat[i-1] + bt[i];
        wtavg = wtavg + wt[i];
        tatavg = tatavg + tat[i];
    }
    printf("\nPROCESS\t\tPRIORITY\tBURST TIME\tWAITING TIME\tTURNAROU
ND TIME");
    for(i = 0; i < n; i++)
        printf("\n%d \t\t %d \t\t %d \t\t %d \t\t %d", p[i], pri[i], bt[i], wt[i], tat[i]);
    printf("\nAverage Waiting Time is --- %f", wtavg / n);
    printf("\nAverage Turnaround Time is --- %f\n", tatavg / n);
```

```
    getchar();
}
```

INPUT:

```
Enter the number of processes --- 4
Enter the Burst Time & Priority of Process 0 --- 10 3
Enter the Burst Time & Priority of Process 1 --- 5 2
Enter the Burst Time & Priority of Process 2 --- 8 1
Enter the Burst Time & Priority of Process 3 --- 6 4
```

OUTPUT:

```
PROCESS  PRIORITY  BURST TIME  WAITING TIME  TURNAROUND TIME
2        1         8           0             8
1        2         5           8             13
0        3         10          13            23
3        4         6           23            29
Average Waiting Time is --- 11.000000
Average Turnaround Time is --- 18.250000
```

This record maintains the structure of your content without modifying the code or explanations.