WEEK 3

Write a C program to simulate the following CPU scheduling algorithm to find turnaround time and waiting time. ?

***Priority (pre-emptive & Non-pre-emptive)**

***Round Robin (Experiment with different quantum sizes for RR algorithm)**

Date : 22/6/23

Write a program to simulate the following CPU scheduling algorithm to find turnaround time and waiting time.
— Priority
— Round Robin

```
#include<stdio.h>
#include<stdlib.h>
void waitingtime(int proc[], int n, int burst_time[],
                 int wait[])
{
    wait_time[0] = 0;
    for(int i = 1; i<n; i++)
        wait_time[i] = wait_time[i-1] + burst_time[i-1];
}
void turnaroundtime(int proc[], int n, int burst_time[],
                    int wait[], int tat[])
{
    for(int i=0; i<n; i++)
        tat[i] = burst_time[i] + wait_time[i];
}
void avgtime(int proc[], int n, int burst_time[])
{
    int wait_time[n], tat[n], total_wt = 0, total_tat = 0;
    waitingtime(proc, n, burst_time, wait_time);
    turnaroundtime(proc, n, burst_time, wait_time, tat);
    for(int i=0; i<n; i++)
    {
        total_wt += wait_time[i];
        total_tat += tat[i];
        printf("%d \t %d \t %d \t %d", proc[i],
          burst_time[i], wait_time[i], tat[i]);
    }
    Printf("\n Average wait time: %d \n
    Average turnaround time: %d", total_wt, total_tat);
}
```

```
void sort(int proc[], int burst_time[], int n, int
        priority[]){
    int a, b, c;
    for(int i=0; i<n; i++){
        for(int j=i+1; j<n; j++){
            if(priority[i] > priority[j]){
                a = burst_time[i];
                burst_time[i] = burst_time[j];
                burst_time[j] = a;
                b = proc[i];
                proc[i] = proc[j];
                proc[j] = b;
                c = priority[i];
                priority[i] = priority[j];
                priority[j] = c;
            }
        }
    }
}
void main()
{
    int proc[10], burst_time[10], n, priority[10];
    printf("\n Enter the size of n:");
    scanf("%d", &n);
    for(int i=0; i<n; i++)
    {
        printf("\n Enter the processor number:");
        scanf("%d", &proc[i]);
        printf("\n Enter the burst time:");
        scanf("%d", &burst_time[i]);
        printf("\n Enter the priority:");
        scanf("%d", &priority[i]);
    }
    sort(proc, burst_time, n, priority);
    avgtime(proc, n, burst_time);
}
```

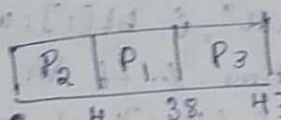
Output:

```
Enter the size of n: 3
Enter the process number: 1
Enter the burst time: 24
Enter the priority: 2
Enter the process number: 2
Enter the burst time: 4
Enter the priority: 1
Enter the process number: 3
Enter the burst time: 5
Enter the priority: 3
```

Process	Burst time	wait time	Turnaround time
2	4	0	4
1	24	4	38
3	5	38	43

Average wait time: 14
Average turnaround time: 28

Gaint Chart:



Round Robin Scheduling

```
#include <stdio.h>
#define MAX_PROCESSES 10
void roundRobin(int n, int bt[], int quantum)
{
  int rem_bt[MAX_PROCESSES], wt[MAX_PROCESSES], i;
  int total_wt=0, total_time=0;
  for(int i=0; i<n; i++)
    rem_bt[i] = bt[i];
  while(1)
  {
    int done = 1;
    for(i=0; i<n; i++) {
```

```
if (rem_bt[i] > 0){
    done = 0;
    if (rem_bt[i] > quantum){
        total_time += quantum;
        rem_bt[i] -= quantum;
    } else {
        total_time += rem_bt[i];
        wt[i] = total_time - bt[i];
        rem_bt[i] = 0
    }
}
}
if(done == 1)
    break;
}

for (i=0 ; i<n ; i++){
    total_wt += wt[i]; total_tat = wt[i] + burst_time[i];}
printf ("Process \t Burst time \t waiting time \n");
for (i=0; i<n; i++)
    printf ("%d \t %d \t %d \n", i, bt[i], wt[i]);
printf ("Average waiting time: %.2f \n, Average
turnaround time: %.2f ", (float)total_wt/n,
(float) total_tat /n);
}

void main()
{
int n, i, bt[MAX_PROCESSES], quantum;
printf ("Enter the number of processes: ");
scanf ("%d", &n);
printf ("Enter burst time for each process: \n");
for (i= 0; i<n; i++){
    printf ("Process %d:", i);
    scanf ("%d", &bt[i]);
}
printf ("Enter quantum size:");
scanf ("%d", &quantum);
```

roundRobin(n, bt, quantum);
}

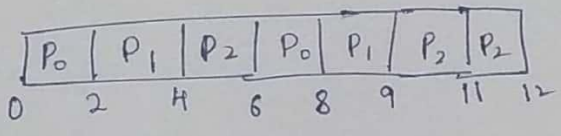
Output:

Enter the number of processes: 3
Enter burst time for each process:
Process 0: 4
Process 1: 3
Process 2: 5
Enter the quantum size: 2

Process	Burst time	Waiting time	Turnaround time
0	4	4	8
1	3	6	9
2	5	7	12

Average waiting time: 5.67
Average turnaround time: 9.67

Gaint chart:



22/6/23

Priority C Program:

```
#include <stdio.h>
#include <stdlib.h>
void waitingtime(int proc[], int n, int burst_time[], int wait_time[])
{
   wait_time[0] = 0;
   for (int i = 1; i < n; i++)
   {
      wait_time[i] = burst_time[i - 1] + wait_time[i - 1];
   }
}
void turnaroundtime(int proc[], int n, int burst_time[], int wait_time[], int tat[])
{
   for (int i = 0; i < n; i++)
      tat[i] = burst_time[i] + wait_time[i];
}
void avgtime(int proc[], int n, int burst_time[])
{
   int wait_time[n], tat[n], total_wt = 0, total_tat = 0;
   waitingtime(proc, n, burst_time, wait_time);
   turnaroundtime(proc, n, burst_time, wait_time, tat);
   printf("\n Process \t  Burst Time \t Wait Time \t  Turnaround time");
   for (int i = 0; i < n; i++)
   {
      total_wt += wait_time[i];
      total_tat += tat[i];
      printf("\n %d\t\t%d\t\t%d\t\t%d", proc[i], burst_time[i], wait_time[i], tat[i]);

}
 printf("\nAverage wait time:%d  Average turnaround time:%d \n", total_wt / n, total_tat / n);
}
```

```
void sort(int proc[],int burst_time[],int n,int priority[]){
   int a,b,c;
   for(int i=0;i<n;i++){
      for(int j=i+1;j<n;j++){
         if(priority[i]>priority[j]){
            // swap priority
            a=burst_time[i];
            burst_time[i]=burst_time[j];
            burst_time[j]=a;
            // swap proc accordingly
            b=proc[i];
            proc[i]=proc[j];
            proc[j]=b;
            // swap priority
            c=priority[i];
            priority[i]=priority[j];
            priority[j]=c;
         }
      }
   }
}
void main()
{
   int proc[10], burst_time[10], n,priority[10];
   printf("\n Enter the size of n:");
   scanf("%d", &n);
   for (int i = 0; i < n; i++)
   {
      printf("Enter the processor number:");
      scanf("%d", &proc[i]);
      printf("Enter the burst time:");
```

```
        scanf("%d", &burst_time[i]);
        printf("enter the priority:");
        scanf("%d",&priority[i]);
    }
    sort(proc,burst_time,n,priority);
    avgtime(proc, n, burst_time);
}
```

OUTPUT:

```
Enter the size of n:3
Enter the processor number:1
Enter the burst time:34
enter the priority:2
Enter the processor number:2
Enter the burst time:4
enter the priority:1
Enter the processor number:3
Enter the burst time:5
enter the priority:3
Process        Burst Time     Wait Time    Turnaround time
 2       4        0        4
 1       34       4        38
 3       5        38       43
Average wait time:14  Average turnaround time:28
```

Round Robin C program:

```
#include<stdio.h>
#define MAX_PROCESSES 10
void roundRobin(int n, int bt[], int quantum) {
    int rem_bt[MAX_PROCESSES], wt[MAX_PROCESSES], i,tat[MAX_PROCESSES];
    int total_wt = 0, total_time = 0,total_tat=0 ;
    for (i = 0; i < n; i++)
        rem_bt[i] = bt[i];
    while (1) {
        int done = 1;
```

```
        for (i = 0; i < n; i++) {
            // If burst time remaining for the process
            if (rem_bt[i] > 0) {
                done = 0; // There is still a pending process
                // If burst time is greater than quantum
                if (rem_bt[i] > quantum) {
                    total_time += quantum;
                    rem_bt[i] -= quantum;
                } else {
                    // Last cycle for this process
                    total_time += rem_bt[i];
                    wt[i] = total_time - bt[i];
                    rem_bt[i] = 0;
                }
            }
        }

        // If all processes are done
        if (done == 1)
            break;
    }
for (i = 0; i < n; i++)
    tat[i]=wt[i]+bt[i];
    // Calculate total waiting time
    for (i = 0; i < n; i++)
    {
        total_wt += wt[i];
        total_tat+=tat[i];
    }

    // Print results
```

```
    printf("Process\tBurst Time\tWaiting Time\tTurnaround time\n");
    for (i = 0; i < n; i++)
        printf("%d\t%d\t\t%d\t%d\n", i, bt[i], wt[i],tat[i]);
    printf("Average waiting time: %.2f\nAverage turnaround time: %.2f\n ", (float)total_wt / n,(float)total_tat / n);
}

int main() {
    int n, i, bt[MAX_PROCESSES], quantum;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    printf("Enter burst time for each process:\n");
    for (i = 0; i < n; i++) {
        printf("Process %d: ", i);
        scanf("%d", &bt[i]);
    }
    printf("Enter the quantum size: ");
    scanf("%d", &quantum);
    roundRobin(n, bt, quantum);
    return 0;
}
```

OUTPUT:

```
Enter the number of processes: 3
Enter burst time for each process:
Process 0: 4
Process 1: 3
Process 2: 5
Enter the quantum size: 2
Process Burst Time  Waiting Time    Turnaround time
0   4       4   8
1   3       6   9
2   5       7   12
Average waiting time: 5.67
Average turnaround time: 9.67
```