Week - 3
(22 June 2023)
Experiment - 3

Question:

Write a C program to simulate the following CPU scheduling algorithm to find turnaround time and waiting time:
(a) Priority (Non-pre-emptive)
(b) Round Robin (Non-pre-emptive)

Program:

(a) Priority (Non-pre-emptive)

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

struct process {
   int process_id;
   int burst_time;
   int priority;
   int waiting_time;
   int turnaround_time;
};

void find_average_time(struct process[], int);

void priority_scheduling(struct process[], int);

int main()
{
   int n, i;
   struct process proc[10];

   printf("Enter the number of processes: ");
   scanf("%d", &n);

for(i = 0; i< n; i++)
   {
      printf("\nEnter the process ID: ");
      scanf("%d", &proc[i].process_id);

      printf("Enter the burst time: ");
      scanf("%d", &proc[i].burst_time);

      printf("Enter the priority: ");
      scanf("%d", &proc[i].priority);
   }

   priority_scheduling(proc, n);
   return 0;
}
```

```
void find_waiting_time(struct process proc[], int n, int wt[])
{
   int i;
   wt[0] = 0;

   for(i = 1; i< n; i++)
   {
      wt[i] = proc[i - 1].burst_time + wt[i - 1];
   }
}

void find_turnaround_time(struct process proc[], int n, int wt[], int tat[])
{
   int i;
   for(i = 0; i< n; i++)
   {
      tat[i] = proc[i].burst_time + wt[i];
   }
}

void find_average_time(struct process proc[], int n)
{
   int wt[10], tat[10], total_wt = 0, total_tat = 0, i;

   find_waiting_time(proc, n, wt);
   find_turnaround_time(proc, n, wt, tat);

   printf("\nProcess ID\tBurst Time\tPriority\tWaiting Time\tTurnaround Time");

   for(i = 0; i< n; i++)
   {
      total_wt = total_wt + wt[i];
      total_tat = total_tat + tat[i];
      printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", proc[i].process_id, proc[i].burst_time, proc[i].priority, wt[i],
               tat[i]);
   }
   printf("\n\nAverage Waiting Time = %f", (float)total_wt/n);
   printf("\nAverage Turnaround Time = %f\n", (float)total_tat/n);
}

void priority_scheduling(struct process proc[], int n)
{
   int i, j, pos;
   struct process temp;
   for(i = 0; i< n; i++)
   {
      pos = i;
      for(j = i + 1; j < n; j++)
      {
         if(proc[j].priority< proc[pos].priority)
         pos = j;
      }
      temp = proc[i];
      proc[i] = proc[pos];
      proc[pos] = temp;
```

```
    }
    find_average_time(proc, n);
}
```

(b) <u>Round Robin (Non-pre-emptive)</u>

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

int turnarroundtime(int processes[], int n, int bt[], int wt[], int tat[]) {
   for (int i = 0; i < n ; i++)
   tat[i] = bt[i] + wt[i];
   return 1;
}

int waitingtime(int processes[], int n, int bt[], int wt[], int quantum)
{
   int rem_bt[n];
   for (int i = 0 ; i < n ; i++)
   rem_bt[i] = bt[i];
   int t = 0;

    while (1)
    {
        bool done = true;

        for (int i = 0 ; i < n; i++)
        {
            if (rem_bt[i] > 0)
            {
                done = false;
                if (rem_bt[i] > quantum)
                {
                    t += quantum;
                    rem_bt[i] -= quantum;
                }

                else
                {
                    t = t + rem_bt[i];
                    wt[i] = t - bt[i];
                    rem_bt[i] = 0;
                }
            }
      }
      if (done == true)
         break;
   }
   return 1;
}

int findavgTime(int processes[], int n, int bt[], int quantum) {
    int wt[n], tat[n], total_wt = 0, total_tat = 0;

    waitingtime(processes, n, bt, wt, quantum);
```

```
    turnarroundtime(processes, n, bt, wt, tat);

    printf("\n\nProcesses\t\t Burst Time\t\t Waiting Time\t\t turnaround time\n");
    for (int i=0; i<n; i++)
    {
        total_wt = total_wt + wt[i];
        total_tat = total_tat + tat[i];
        printf("\n\t%d\t\t\t%d\t\t\t%d\t\t\t%d\n",i+1, bt[i], wt[i], tat[i]);
    }

    printf("\nAverage waiting time = %f", (float)total_wt / (float)n);
    printf("\nAverage turnaround time = %f", (float)total_tat / (float)n);
    return 1;
}

int main()
{
   int n, processes[n], burst_time[n], quantum;
   printf("Enter the Number of Processes: ");
   scanf("%d",&n);

   printf("\nEnter the quantum time: ");
   scanf("%d",&quantum);

    int i=0;
    for(i=0;i<n;i++)
    {
        printf("\nEnter the process: ");
        scanf("%d",&processes[i]);
        printf("Enter the Burst Time:");
        scanf("%d",&burst_time[i]);
    }

   findavgTime(processes, n, burst_time, quantum);
   return 0;
}
```

Output:

(a) Priority (Non-pre-emptive)

```
Enter the number of processes: 3

Enter the process ID: 1
Enter the burst time: 10
Enter the priority: 3

Enter the process ID: 2
Enter the burst time: 8
Enter the priority: 2

Enter the process ID: 3
Enter the burst time: 5
Enter the priority: 1

Process ID      Burst Time      Priority        Waiting Time    Turnaround Time
3               5               1               0               5
2               8               2               5               13
1               10              3               13              23

Average Waiting Time = 6.000000
Average Turnaround Time = 13.666667
```

(b) Round Robin (Non-pre-emptive)

```
Enter the Number of Processes: 3

Enter the quantum time: 2

Enter the process: 1
Enter the Burst Time:4

Enter the process: 2
Enter the Burst Time:3

Enter the process: 3
Enter the Burst Time:5


Processes               Burst Time              Waiting Time            turnaround time

        1                       4                       4                       8

        2                       3                       6                       9

        3                       5                       7                       12

Average waiting time = 5.666667
Average turnaround time = 9.666667
```

Observation Book Pictures:

PAGE NO :
DATE : 22/06/23

Experiment - 3

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

(a) Priority (Non-pre-emptive)
(b) Round Robin

Program:

(a) Priority:

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

struct process{
 int process_id;
 int burst_time;
int priority;
 int waiting_time;
 int turnaround_time;
};

void find_average_time (struct process[], int);
void priority_scheduling(struct process[], int);

int main()
{ int n, i;
   struct process proc[10];
    printf("Enter the number of processes: ");
    scanf("%d", &n);
```

→

```
for (i=0; i<n; i++)
  { printf ("Enter the process ID: ");
    scanf ("%d", &proc[i].process_id);

    printf ("Enter the burst time: ");
    scanf ("%d", &proc[i].burst_time);

    printf ("Enter the priority: ");
    scanf ("%d", &proc[i].priority);
  }
priority_scheduling (proc, n);
return 0;
}

Void find_waiting_time (struct process proc[], int n,
                        int wt[])
{ int i;
  wt[0] = 0
  for (i=1; i<n; i++)
    { wt[i] = proc[i-1].burst_time + wt[i-1]
    }
}

Void find_turnaround_time (struct process proc[],
                           int n, int wt[], int tat[])
{ int i;
  for (i=0; i<n; i++)
    { tat[i] = proc[i].burst_time + wt[i];
    }
}
```

```
void find_average_time (struct process proc[], int n)
{ int wt[10], tat[10], total_wt = 0, total_tat = 0, i;

    find_waiting_time (proc, n, wt);
    find_turnaround_time (proc, n, wt, tat);

printf("\n Process ID\t Burst Time \t Priority \t
        Waiting Time \t Turnaround Time ");

  for (i=0; i<n; i++)
    { total_wt = total_wt + wt[i];
      total_tat = total_tat + tat[i];
    printf("\n%d \t \t%d \t \t %d \t \t %d ",
          proc[i].process_id, proc[i].burst_time,
          proc[i].priority, wt[i], tat[i]);
    }

printf("\n\nAverage Waiting Time = %f ",
          (float) total_wt/n);
printf("\n Average Turnaround Time = %f\n",
          (float) total_tat/n);

void priority_scheduling (struct process proc[],
                          int n)
{ int i, j, pos;
  struct process temp;
  for (i=0; i<n; i++)
    { pos=i;
      for (j=i+1; j<n; j++)
        { if (proc[j].priority < proc[pos].priority)
          pos=j;
        }
```

→

```
        temp = proc[i];
        proc[i] = proc[pos];
        proc[pos] = temp;
    }
    find_average_time(proc, n);
}
```

Output:

Enter the number of processes : 3

Enter the process ID = 1
Enter the burst time = 10
Enter the priority = 3

Enter the process ID = 2
Enter the ~~process~~ burst time = 8
Enter the priority = 2

Enter the process ID = 3
Enter the burst time = 5
Enter the priority = 1

Process ID	Burst Time	Priority	Waiting Time	Turnaround Time
3	5	1	0	5
2	8	2	5	13
1	10	3	13	23

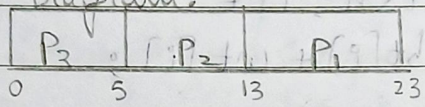
Average Waiting Time = 6.000
Average Turnaround Time = 13.666

PAGE NO :
DATE :

Traversal:

Processes	Burst time	Priority
P_1	10	3
P_2	8	2
P_3	5	1

Grant Diagram:



Waiting Time	Turnaround Time	Process ID
0	5	P3
5	13	P2
13	23	P1

Avg waiting Time = $\frac{0+5+13}{3}$ = 6.0000

Avg Turnaround Time = $\frac{5+13+23}{3}$ = 13.6666.

PAGE NO :
DATE : 22/06/23

(b) Round Robin

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

int turnaroundtime (int processes, int n, int bt[],
                    int wt[], int tat[])
{  for (int i=0; i<n; i++
     { tat[i] = bt[i] + wt[i];
     }

   return 1;
}

int waitingtime (int processes[], int n, int bt[],
                 int wt[], int tat[])
{ int rem_bt[n];
  for (int i=0; i<n; i++)
  rem_bt[i] = bt[i];
  int t=0;

  while (1)
  { bool done = true;
    for (int i=0; i<n; i++)
     { if (rem_bt[i] > 0)
        { done = false;
          if (rem_bt[i] > quantum)
           { t += quantum;
             rem_bit[i] -= quantum;
           }
        else
          {t = t + rem_bt[i];
           wt[i] = t - bt[i];
           rem_bt[i] = 0;
          }
```

```
            }
        }
        if (done == true)
            break;
    }
    return 1;
}

int findavgTime (int processes[], int n, int bt[],
                 int quantum)
{ int wt[n], tat[n], total_wt = 0, total_tat = 0;

  waitingtime (processes, n, bt, wt, quantum);
  turnaroundtime (processes, n, bt, wt, tat);

  printf ("\n\n Processes \t\t Burst Time \t\t
          Waiting Time \t\t Turnaround Time \n");
  for (int i = 0; i<n; i++)
    { total_wt = total_wt + wt[i];
      total_tat = total_tat + tat[i];
      printf ("\n\t %d \t\t\t %d \t\t\t %d \t\t\t %d\n",
              i+1, bt[i], wt[i], tat[i]);
    }
  printf ("\n Average waiting Time = %f ",
          (float) total_wt / (float) n);
  printf ("\n Average Turnaround Time = %f ",
          (float) total_tat / (float) n);
  return 1;
}
```

```
int main()
{ int n, processes[n], burst_time[n], quantum;
  printf("Enter the Number of Processes: ");
  scanf("%d", &n);

  printf("Enter the quantum time: ");
  scanf("%d", &quantum);

  int i = 0;
  for (i=0; i<n; i++)
    { printf("\n Enter the process: ");
      scanf("%d", &processes[i]);
      printf("Enter the Burst time: ");
      scanf("%d", &burst_time[i]);
    }

  findavgTime(processes, n, burst_time, quantum);
  return 0;
}
```

Output:

```
Enter the Number of Processes: 3

Enter the quantum time: 2

Enter the process: 1
Enter the Burst Time: 4

Enter the process: 2
Enter the Burst Time: 3

Enter the process: 3
Enter the Burst Time: 5
```

PAGE NO :
DATE :

Processes	Burst Time	waiting Time	Turnaround Time
1	4	4	8
2	3	6	9
3	5	7	12

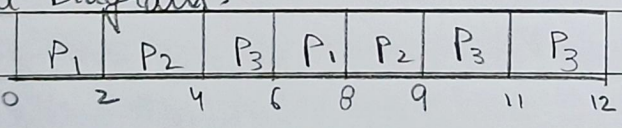
Average Waiting time : 5.6666
Average Turnaround time : 9.6666

Traversal:

Process	Burst time	waiting Time	Turnaround Time
P1	4	4	8
P2	3	6	9
P3	5	7	12

Quantum Time : 2

Grant Diagram:



Avg waiting Time : $\frac{4+6+7}{3} = 5.6666$

Avg Turnaround Time : $\frac{8+9+12}{3} = 9.6666$

23/6/23