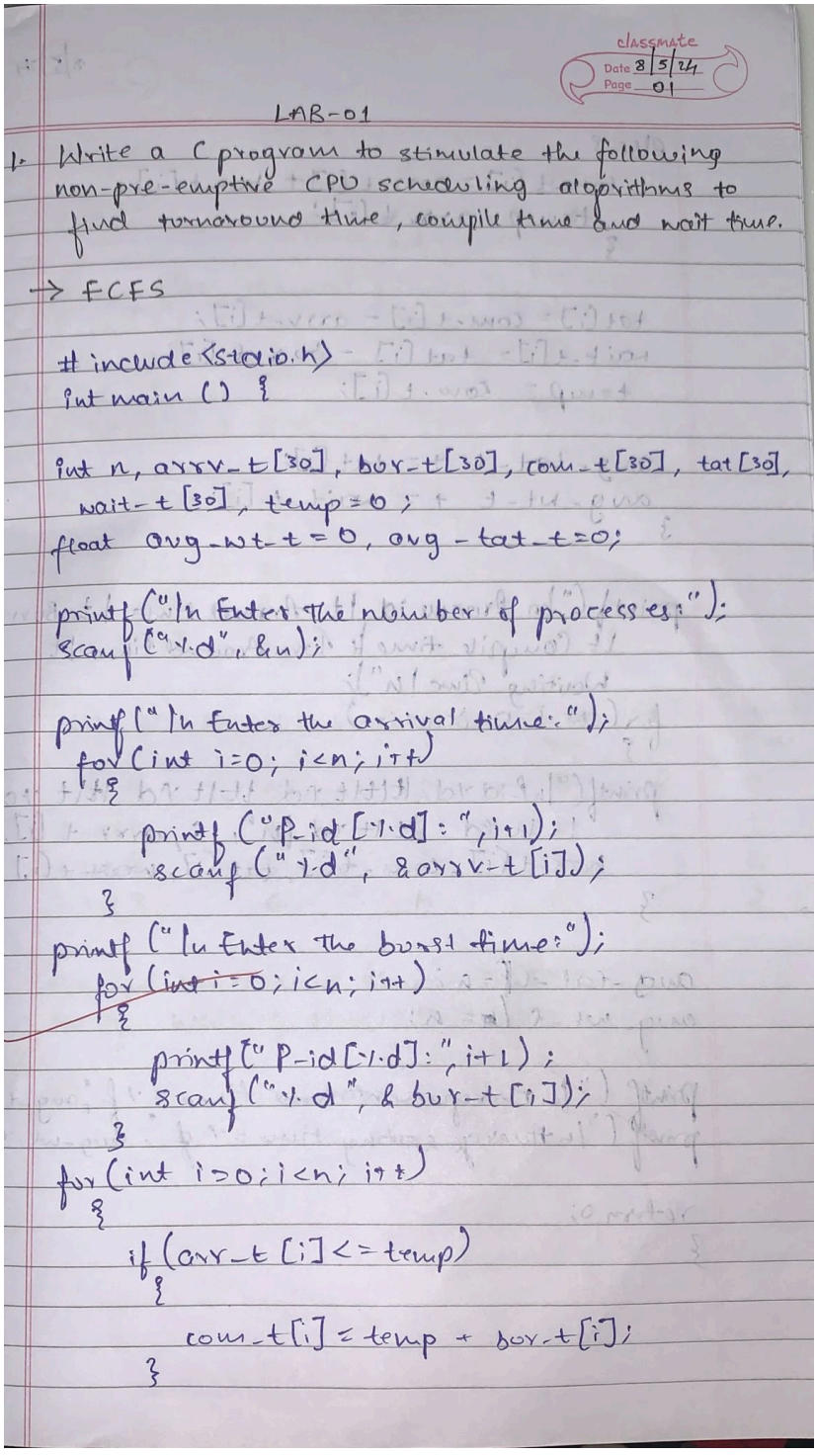
Operating System Lab Observation

Mohith Jain 1BM22CS162

Lab 1:



LAB-01

1. Write a C program to simulate the following non-pre-emptive CPU scheduling algorithms to find turnaround time, compile time and wait time.

→ FCFS

```
#include<stdio.h>
int main () {

int n, arrv_t[30], bur_t[30], com_t[30], tat[30],
  wait_t[30], temp=0;
float avg_wt_t=0, avg_tat_t=0;

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

printf("\n Enter the arrival time:");
for(int i=0; i<n; i++)
{
   printf("P_id[%d]:", i+1);
   scanf("%d", &arrv_t[i]);
}
printf("\n Enter the burst time:");
for(int i=0; i<n; i++)
{
   printf("P_id[%d]:", i+1);
   scanf("%d", &bur_t[i]);
}
for(int i=0; i<n; i++)
{
   if(arr_t[i]<=temp)
   {
      com_t[i] = temp + bur_t[i];
   }
```

```
else {
    com_t[i] = arr_t[i] + bur_t[i];
    temp = arr_t[i];
    }

    tat[i] = com_t[i] - arr_t[i];
    wait_t[i] = tat[i] - bur_t[i];
    temp = com_t[i];

    avg_tat_t += tat[i];
    avg_wt_t += wait_t[i];
}

printf("\n Process id \t Arrival time \t Burst time
    \t Compile time \t Turn Around time \t
    Waiting time \n");
for(int i=0; i<n; i++)
{
printf("\n P_id %d \t\t %d \t\t %d \t\t %d \t\t %d
    \t\t %d \n", i, arr_t[i],
    bur_t[i], com_t[i], tat[i], wait_t[i]);
}

avg_tat_t /= n;
avg_wt_t /= n;

printf("\n Average turnaround time = %f", avg_tat_t);
printf("\n Average waiting time = %f", avg_wt_t);

return 0;
}
```

Output:

```
Enter the number of process: 4

Enter the Arrival time:
P_id[1] : 0
P_id[2] : 1
P_id[3] : 5
P_id[4] : 6

Enter the Burst Time:
P_id[1] : 2
P_id[2] : 2
P_id[3] : 3
P_id[4] : 4
```

Process id	Arrival time	Burst time	Compile time	TAT time	Waiting time
P_id 0	0	2	2	2	0
P_id 1	1	2	4	3	1
P_id 2	5	3	8	3	0
P_id 3	6	4	12	6	2

```
Average turnaround time = 3.500000
Average waiting time = 0.750000
```

SJF

```
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

void sort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                swap(&arr[j], &arr[j + 1]);
            }
        }
    }
}

int main() {
    int n, arrv_t[30], bur_t[30], com_t[30], wait_t[30], tat[30], temp = 0;
    float avg_wt_t = 0, avg_tat_t = 0;

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

    printf("\nEnter the arrival time:");
    for (int i = 0; i < n; i++) {
        printf("P_id[%d]:", i + 1);
        scanf("%d", &arrv_t[i]);
    }

    printf("\nEnter the burst time:");
    for (int i = 0; i < n; i++) {
        printf("P_id[%d]:", i + 1);
        scanf("%d", &bur_t[i]);
    }

    sort(bur_t, n); // Sort burst times

    for (int i = 0; i < n; i++) {
        if (arrv_t[i] <= temp) {
            com_t[i] = temp + bur_t[i];
        } else {
            com_t[i] = arrv_t[i] + bur_t[i];
            temp = arrv_t[i];
        }

        tat[i] = com_t[i] - arrv_t[i];
        wait_t[i] = tat[i] - bur_t[i];
        temp = com_t[i];

        avg_tat_t += tat[i];
        avg_wt_t += wait_t[i];
    }
```

```
    printf("\nProcess id \t Arrival time \t Burst time \t Compile time\t TurnAround time \t
Waiting time\n");
    for (int i = 0; i < n; i++) {
        printf("\nP_id%d \t\t\t %d \t\t %d \t\t %d \t\t %d \t\t %d\n", i, arrv_t[i],
bur_t[i], com_t[i], tat[i], wait_t[i]);
    }

    avg_tat_t /= n;
    avg_wt_t /= n;

    printf("\n Average turnAround time = %f", avg_tat_t);
    printf("\n Average waiting time = %f", avg_wt_t);

    return 0;
}
```

```
Enter the number of processes:4

Enter the arrival time:P_id[1]:0
P_id[2]:1
P_id[3]:5
P_id[4]:6

Enter the burst time:P_id[1]:2
P_id[2]:2
P_id[3]:3
P_id[4]:4

Process id       Arrival time    Burst time      Compile time    TurnAround time  Waiting time

P_id0                    0              2              2              2       0

P_id1                    1              2              4              3       1

P_id2                    5              3              8              3       0

P_id3                    6              4              12             6       2

 Average turnAround time = 3.500000
 Average waiting time = 0.750000%
mohithjain@Mohiths-MacBook-Air Downloads %
```

LAB 02

Priority Scheduling Non-preemptive

```
#include<stdio.h>

void swap(int *a, int *b)
{
    int temp = *a;
    *a= *b;
    *b = temp;
}
void sort(int arr[],int prior[],int n)
{
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (prior[j] > prior[j + 1]) {
                swap(&prior[j], &prior[j + 1]);
                swap(&arr[j], &arr[j + 1]);
            }
        }
    }
}

int main()
{
 int n,prior[30],burt[30],tat[30],waitt[30],temp;
 float avg_waitt=0,avg_tat=0;

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


 for(int i=0;i<n;i++)
 {
      printf("\nEnter the burst time & Priority of process %d:",i);
      scanf("%d %d",&burt[i],&prior[i]);
 }

 sort(burt,prior,n);
  for (int i = 0; i < n; i++) {
        waitt[i] = temp;
        tat[i] = waitt[i] + burt[i];
        temp += burt[i];

        avg_tat += tat[i];
        avg_waitt += waitt[i];
    }

    printf("\nProcess id \t Burst time \t Priority \t TurnAround time \t Waiting time\n");
    for (int i = 0; i < n; i++) {
        printf("\nP_id%d \t\t\t %d \t\t %d \t\t %d \t\t\t %d\n", i, burt[i], prior[i],
tat[i], waitt[i]);
    }

    avg_tat /= n;
```

```
    avg_waitt /= n;

    printf("\n Average turnAround time = %f", avg_tat);
    printf("\n Average waiting time = %f", avg_waitt);

    return 0;
}
```

```
Enter the number of processes:5

Enter the burst time & Priority of process 0:10 3

Enter the burst time & Priority of process 1:1 1

Enter the burst time & Priority of process 2:2 4

Enter the burst time & Priority of process 3:1 5

Enter the burst time & Priority of process 4:5 2

Process id       Burst time      Priority        TurnAround time         Waiting time

P_id0                    1              1               1                       0

P_id1                    5              2               6                       1

P_id2                    10             3               16                      6

P_id3                    2              4               18                      16

P_id4                    1              5               19                      18

 Average turnAround time = 12.000000
 Average waiting time = 8.200000%
mohithjain@Mohiths-MacBook-Air Downloads %
```

Priority Scheduling Preemptive

```
#include <stdio.h>

#define MAX 30

void sort_by_priority(int n, int bur_t[], int pri[], int id[]) {
    int temp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (pri[i] > pri[j]) {
                temp = pri[i];
                pri[i] = pri[j];
                pri[j] = temp;

                temp = bur_t[i];
                bur_t[i] = bur_t[j];
                bur_t[j] = temp;

                temp = id[i];
                id[i] = id[j];
                id[j] = temp;
            }
        }
    }
}
int main() {
    int n;
    int bur_t[MAX], rem_bur_t[MAX], pri[MAX], id[MAX];
    int tat[MAX], wait_t[MAX];
    float avg_waiting_time = 0, avg_turnaround_time = 0;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        id[i] = i + 1;
        printf("Enter burst time of process P%d: ", id[i]);
        scanf("%d", &bur_t[i]);
        rem_bur_t[i] = bur_t[i];
        printf("Enter priority of process P%d: ", id[i]);
        scanf("%d", &pri[i]); }
    sort_by_priority(n, bur_t, pri, id);
    int current_time = 0, completed = 0;
    while (completed != n) {
        int idx = -1;
        int highest_priority = 10000; // assuming a large number as the initial highest
priority
        for (int i = 0; i < n; i++) {
            if (rem_bur_t[i] > 0) {
                if (pri[i] < highest_priority) {
                    highest_priority = pri[i];
                    idx = i;
                }
                if (pri[i] == highest_priority) {
                    if (id[i] < id[idx]) {
                        highest_priority = pri[i];
                        idx = i;
                    }
```

```
                }
            }
        }
        if (idx != -1) {
            current_time++;
            rem_bur_t[idx]--;
            if (rem_bur_t[idx] == 0) {
                completed++;
                tat[idx] = current_time;
                wait_t[idx] = tat[idx] - bur_t[idx];
                avg_waiting_time += wait_t[idx];
                avg_turnaround_time += tat[idx];
            }
        } else {
            current_time++;
        }
    }
    printf("\nProcess ID\tBurst Time\tPriority\tTurnaround Time\tWaiting Time\n");
    for (int i = 0; i < n; i++) {
        printf("P%d\t\t%d\t\t%d\t\t%d\t\t%d\n", id[i], bur_t[i], pri[i], tat[i],
wait_t[i]);
    }
    avg_waiting_time /= n;
    avg_turnaround_time /= n;
    printf("\nAverage Turnaround Time = %.2f", avg_turnaround_time);
    printf("\nAverage Waiting Time = %.2f\n", avg_waiting_time);
    return 0;
}
```

```
Enter the number of processes: 5
Enter burst time of process P1: 10
Enter priority of process P1: 3
Enter burst time of process P2: 1
Enter priority of process P2: 1
Enter burst time of process P3: 2
Enter priority of process P3: 4
Enter burst time of process P4: 1
Enter priority of process P4: 5
Enter burst time of process P5: 5
Enter priority of process P5: 2

Process ID      Burst Time      Priority        Turnaround Time Waiting Time
P2              1               1               10              9
P5              5               2               11              6
P1              10              3               13              3
P3              2               4               14              12
P4              1               5               19              18

Average Turnaround Time = 13.40
Average Waiting Time = 9.60
mohithjain@Mohiths-MacBook-Air Lab %
```

Round Robin

```
#include <stdio.h>

int main() {
    int n, i, tq, time = 0;
    int bur_t[30], rem_bur_t[30], wait_t[30], tat[30];
    float avg_wt_t = 0, avg_tat_t = 0;

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

    printf("\nEnter the burst time for each process:\n");
    for (i = 0; i < n; i++) {
        printf("Burst time of process P%d: ", i + 1);
        scanf("%d", &bur_t[i]);
        rem_bur_t[i] = bur_t[i];
    }

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

    int done = 0;
    while (done != n) {
        for (i = 0; i < n; i++) {
            if (rem_bur_t[i] > 0) {
                if (rem_bur_t[i] > tq) {
                    time += tq;
                    rem_bur_t[i] -= tq;
                } else {
                    time += rem_bur_t[i];
                    wait_t[i] = time - bur_t[i];
                    tat[i] = time;
                    rem_bur_t[i] = 0;
                    done++;
                }
            }
        }
    }

    printf("\nProcess ID\tBurst Time\tTurnaround Time\tWaiting Time\n");
    for (i = 0; i < n; i++) {
        avg_wt_t += wait_t[i];
        avg_tat_t += tat[i];
        printf("P%d\t\t%d\t\t%d\t\t%d\n", i + 1, bur_t[i], tat[i], wait_t[i]);
    }

    avg_wt_t /= n;
    avg_tat_t /= n;

    printf("\nAverage Turnaround Time = %.2f", avg_tat_t);
    printf("\nAverage Waiting Time = %.2f\n", avg_wt_t);

    return 0;
}
```

```
Enter the number of processes: 3

Enter the burst time for each process:
Burst time of process P1: 24
Burst time of process P2: 3
Burst time of process P3: 3

Enter the time quantum: 3

Process ID      Burst Time      Turnaround Time Waiting Time
P1              24              30              6
P2              3               6               3
P3              3               9               6

Average Turnaround Time = 15.00
Average Waiting Time = 5.00
mohithjain@Mohiths-MacBook-Air T %
```