Innovative Assignment

Name: Namit Patel, Kush Patel

Roll No.: 21BCE209, 21BCE205

Batch: C2

Subject: Operating System (2CS403)

//code

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

int count_processes(const char *filename)
{
    FILE *file = fopen(filename, "r");
    int count = 1;
    char c;

    if (file)
    {
        while ((c = getc(file)) != EOF)
        {
            if (c == '\n')
            {
                count++;
            }
        }

        fclose(file);
    }
    else
```

```
    {
        printf("File not found!\n");
    }

    return count;
}

void FCFS()
{
    int n;
    int time = 0;
    char time_unit[7];

    float avg_waiting = 0, avg_tat = 0;
    FILE *fp;
    fp = fopen("input.txt", "r");
    if (fp == NULL)
    {
        printf("Error opening file!!!");
    }
    else
    {
        n = count_processes("input.txt") - 1;
        fscanf(fp, "%s", &time_unit);
        int process[n];
        int waiting[n], turnaround[n], arrival[n],
burst[n];

        for (int a = 0; a < n; a++)
        {
            fscanf(fp, "%d %d", &arrival[a],
&burst[a]);
        }
        fclose(fp);
        waiting[0] = 0;
```

```
        turnaround[0] = burst[0];
        fp = fopen("output.txt", "w");
        for (int a = 1; a < n; a++)
        {
            int sum = 0;
            for (int b = 0; b < a; b++)
                sum += burst[b];
            waiting[a] = sum - arrival[a];
            if (waiting[a] < 0)
                waiting[a] = 0;
            turnaround[a] = waiting[a] + burst[a];
        }

        for (int a = 0; a < n; a++)
        {
            avg_waiting += waiting[a];
            avg_tat += turnaround[a];
        }
        avg_waiting /= n;
        avg_tat /= n;
        fprintf(fp, "Average Waiting Time is: %0.2f
%s", avg_waiting, time_unit);
        fprintf(fp, "\nAverage Turnaround Time: %0.2f
%s", avg_tat, time_unit);
        fprintf(fp, "\nGantt Chart:\n");
        fprintf(fp, "|");
        for (int a = 0; a < n; a++)
        {
            for (int b = 0; b < burst[a]; b++)
            {
                fprintf(fp, "P%d|", a + 1);
                time++;
            }
        }
        fclose(fp);
```

```
    }
    printf("\n");
}

void SJF()
{
    int n, smallest, time = 0, completed = 0;
    char time_unit[7];
    float avg_waiting = 0, avg_tat = 0;
    FILE *fp;
    fp = fopen("input.txt", "r");
    if (fp == NULL)
    {
        printf("Error opening file!!!");
    }
    else
    {
        n = count_processes("input.txt") - 1;
        fscanf(fp, "%s", &time_unit);
        int burst[n], waiting[n], turnaround[n],
arrival[n], remaining[n];

        for (int a = 0; a < n; a++)
        {
            fscanf(fp, "%d %d", &arrival[a],
&burst[a]);
            remaining[a] = burst[a];
        }
        fclose(fp);
        fp = fopen("output.txt", "w");
        fprintf(fp, "Gantt Chart:\n");
        while (completed != n)
        {
            smallest = -1;
            for (int a = 0; a < n; a++)
```

```
            {
                if (remaining[a] > 0 && (smallest ==
-1 || remaining[a] < remaining[smallest]) &&
arrival[a] <= time)
                {
                    smallest = a;
                }
            }
            if (smallest == -1)
            {
                fprintf(fp, "| idle ");
                time++;
            }
            else
            {
                fprintf(fp, "| P%d ", smallest + 1);
                remaining[smallest]--;
                time++;
                if (remaining[smallest] == 0)
                {
                    completed++;
                    waiting[smallest] = time -
arrival[smallest] - burst[smallest];
                    if (waiting[smallest] < 0)
                        waiting[smallest] = 0;
                    turnaround[smallest] =
waiting[smallest] + burst[smallest];
                }
            }
        }
        fprintf(fp, "|\n");
        for (int a = 0; a < n; a++)
        {
            avg_waiting += waiting[a];
            avg_tat += turnaround[a];
```

```
        }
        fprintf(fp, "Average Waiting Time: %.2f %s\n",
avg_waiting / n, time_unit);
        fprintf(fp, "Average Turnaround Time: %.2f
%s\n", avg_tat / n, time_unit);
        fclose(fp);
    }
    printf("\n");
}

void RRS()
{
    int n;
    float avg_waiting = 0, avg_tat = 0;
    char time_unit[7];
    FILE *fp;

    fp = fopen("input.txt", "r");
    if (fp == NULL)
    {
        printf("Error opening file!!!");
    }
    else
    {
        n = count_processes("input.txt") - 1;
        fscanf(fp, "%s", &time_unit);
        printf("Number of processes: %d\n", n);
        int burst[n], waiting[n], turnaround[n],
arrival[n], time_quantum, remaining[n], time = 0,
completed = 0, flag;
        for (int a = 0; a < n; a++)
        {
            fscanf(fp, "%d %d", &arrival[a],
&burst[a]);
```

```
            printf("Process %d Arrival Time: %d Burst
Time: %d\n", a + 1, arrival[a], burst[a]);
            remaining[a] = burst[a];
        }
        fclose(fp);
        fp = fopen("output.txt", "w");
        printf("Enter time quantum: ");
        scanf("%d", &time_quantum);
        fprintf(fp, "Gantt Chart:\n");
        while (completed != n)
        {
            flag = 0;
            for (int a = 0; a < n; a++)
            {
                if (remaining[a] > 0 && arrival[a] <=
time)
                {
                    if (remaining[a] <= time_quantum)
                    {
                        fprintf(fp, "| P%d ", a + 1);
                        time += remaining[a];
                        remaining[a] = 0;
                        completed++;
                        waiting[a] = time - arrival[a]
- burst[a];
                        if (waiting[a] < 0)
                            waiting[a] = 0;
                        turnaround[a] = waiting[a] +
burst[a];
                    }
                    else
                    {
                        fprintf(fp, "| P%d ", a + 1);
                        remaining[a] -= time_quantum;
                        time += time_quantum;
```

```
                    }
                    flag = 1;
                }
            }
            if (flag == 0)
            {
                fprintf(fp, "| idle ");
                time++;
            }
        }
        fprintf(fp, "|\n");
        for (int a = 0; a < n; a++)
        {
            avg_waiting += waiting[a];
            avg_tat += turnaround[a];
        }
        fprintf(fp, "Average Waiting Time: %.2f %s\n",
avg_waiting / n, time_unit);
        fprintf(fp, "Average Turnaround Time: %.2f
%s\n", avg_tat / n, time_unit);
        fclose(fp);
    }
    printf("\n");
}

void PS()
{
    int n;
    float avg_waiting = 0, avg_tat = 0;
    char time_unit[7];
    FILE *fp;
    fp = fopen("priority.txt", "r");
    if (fp == NULL)
    {
        printf("Error opening file!!!");
```

```
    }
    else
    {
        n = count_processes("input.txt") - 1;
        fscanf(fp, "%s", &time_unit);
        int burst[n], priority[n], waiting[n],
turnaround[n], arrival[n], temp, min, time = 0,
completed = 0;
        for (int a = 0; a < n; a++)
        {
            fscanf(fp, "%d %d %d", &arrival[a],
&burst[a], &priority[a]);
        }
        fclose(fp);
        fp = fopen("output.txt", "w");

        for (int a = 0; a < n - 1; a++)
        {
            min = a;
            for (int b = a + 1; b < n; b++)
            {
                if (priority[b] < priority[min])
                    min = b;
            }
            temp = priority[a];
            priority[a] = priority[min];
            priority[min] = temp;
            temp = burst[a];
            burst[a] = burst[min];
            burst[min] = temp;
            temp = arrival[a];
            arrival[a] = arrival[min];
            arrival[min] = temp;
        }
        fprintf(fp, "Gantt Chart:\n");
```

```
        fprintf(fp, "0");
        for (int a = 0; a < n; a++)
        {
            time += burst[a];
            turnaround[a] = time - arrival[a];
            waiting[a] = turnaround[a] - burst[a];
            avg_waiting += waiting[a];
            avg_tat += turnaround[a];
            fprintf(fp, " | P%d | %d", a + 1, time);
        }
        fprintf(fp, "\n");

        fprintf(fp, "Average Waiting Time: %.2f %s\n",
avg_waiting / n, time_unit);
        fprintf(fp, "Average Turnaround Time: %.2f
%s\n", avg_tat / n, time_unit);
        fclose(fp);
    }
    printf("\n");
}

void SRTF()
{
    int n;
    float avg_waiting = 0, avg_tat = 0;
    char time_unit[7];
    FILE *fp;
    fp = fopen("input.txt", "r");
    if (fp == NULL)
    {
        printf("Error opening file!!!");
    }
    else
    {
        n = count_processes("input.txt") - 1;
```

```
        int burst[n], remaining[n], waiting[n],
turnaround[n], arrival[n], time = 0, completed = 0,
min_index;
        fscanf(fp, "%s", &time_unit);
        for (int a = 0; a < n; a++)
        {
            fscanf(fp, "%d %d", &arrival[a],
&burst[a]);
            remaining[a] = burst[a];
        }
        fclose(fp);
        fp = fopen("output.txt", "w");

        fprintf(fp, "Gantt Chart:\n");
        fprintf(fp, "0");
        while (completed != n)
        {
            min_index = -1;
            for (int a = 0; a < n; a++)
            {
                if (arrival[a] <= time && remaining[a]
> 0)
                {
                    if (min_index == -1)
                    {
                        min_index = a;
                    }
                    else if (remaining[a] <
remaining[min_index])
                    {
                        min_index = a;
                    }
                }
            }
            if (min_index != -1)
```

```
            {
                remaining[min_index]--;
                time++;
                if (remaining[min_index] == 0)
                {
                    completed++;
                    turnaround[min_index] = time -
arrival[min_index];
                    waiting[min_index] =
turnaround[min_index] - burst[min_index];
                    avg_waiting += waiting[min_index];
                    avg_tat += turnaround[min_index];
                    fprintf(fp, " | P%d | %d",
min_index + 1, time);
                }
            }
            else
            {
                time++;
            }
        }
        fprintf(fp, "\n");

        fprintf(fp, "Average Waiting Time: %.2f %s\n",
avg_waiting / n, time_unit);
        fprintf(fp, "Average Turnaround Time: %.2f
%s\n", avg_tat / n, time_unit);
        fclose(fp);
    }
    printf("\n");
}

void LRTF()
{
    int n;
```

```
    float avg_waiting = 0, avg_tat = 0;
    char time_unit[7];
    FILE *fp;
    fp = fopen("input.txt", "r");
    if (fp == NULL)
    {
        printf("Error opening file!!!");
    }
    else
    {
        n = count_processes("input.txt") - 1;
        int burst[n], remaining[n], waiting[n],
turnaround[n], arrival[n], time = 0, completed = 0,
max_index, max_value = -1;
        fscanf(fp, "%s", &time_unit);
        for (int a = 0; a < n; a++)
        {
            fscanf(fp, "%d %d", &arrival[a],
&burst[a]);
            remaining[a] = burst[a];
        }
        fclose(fp);
        fp = fopen("output.txt", "w");

        fprintf(fp, "Gantt Chart:\n");
        fprintf(fp, "0");
        while (completed != n)
        {
            max_value = -1;
            for (int a = 0; a < n; a++)
            {
                if (arrival[a] <= time && remaining[a]
> max_value)
                {
                    max_value = remaining[a];
```

```
                    max_index = a;
                }
            }
            if (max_value != -1)
            {
                remaining[max_index]--;
                time++;
                if (remaining[max_index] == 0)
                {
                    completed++;
                    turnaround[max_index] = time -
arrival[max_index];
                    waiting[max_index] =
turnaround[max_index] - burst[max_index];
                    avg_waiting += waiting[max_index];
                    avg_tat += turnaround[max_index];
                    fprintf(fp, " | P%d | %d",
max_index + 1, time);
                }
            }
            else
            {
                time++;
            }
        }
        fprintf(fp, "\n");

        fprintf(fp, "Average Waiting Time: %.2f %s\n",
avg_waiting / n, time_unit);
        fprintf(fp, "Average Turnaround Time: %.2f
%s\n", avg_tat / n, time_unit);
        fclose(fp);
    }
    printf("\n");
}
```

```
int main()
{
    int choice;
st:
    printf("1. First-Come First-Served \n2. Shortest
Job First \n3. Round Robin Scheduling \n4. Priority
Scheduling\n5. Shortest Remaining Time First \n6.
Longest remaining time first \n7. Exit");
    printf("\nEnter choice: ");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
    {

        FCFS();
        goto st;
    }
    case 2:
    {

        SJF();
        goto st;
    }
    case 3:
    {

        RRS();
        goto st;
    }
    case 4:
    {

        PS();
        goto st;
```

```
    }
    case 5:
    {

        SRTF();
        goto st;
    }
    case 6:
    {

        LRTF();
        goto st;
    }
    case 7:
    {
        return 0;
    }
    default:
    {
        printf("Enter valid choice!!!");
        goto st;
    }
    }
}
```

// Input Files

OS_innovative.c input.txt

final > input.txt

```
ms
0 5
1 3
2 1
3 2
4 3
```

// for priority scheduling

OS_innovative.c priority.txt

final > priority.txt

```
ms
0 4 1
0 3 2
6 7 1
11 4 3
12 2 2
```

// Output Screenshots

FCFS

```
Average Waiting Time is: 4.60 ms
Average Turnaround Time: 7.40 ms
Gantt Chart:
|P1|P1|P1|P1|P1|P2|P2|P2|P3|P4|P4|P5|P5|P5|
```

```
https://aka.ms/PowerShell-Release?tag=v7.3.3

PS D:\nirma\innovatiove\OS>  & 'c:\Users\DELL\.vscode\extensions\ms-vscode.cpptools-1.14.5-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Microsoft-MIEngine-In-lc50vklk.oku' '--stdout=Microsoft-MIEngine-Out-tza3wmfu.mwm' '--stderr=Microsoft-MIEngine-Error-fw2rw215.wsf' '--pid=Microsoft-MIEngine-Pid-52fkuha1.4qo' '--dbgExe=C:\mingw32\bin\gdb.exe' '--interpreter=mi'
1. First-Come First-Served
2. Shortest Job First
3. Round Robin Scheduling
4. Priority Scheduling
5. Shortest Remaining Time First
6. Longest remaining time first
7. Exit
Enter choice: 1
```

SJF

```
Gantt Chart:
| P1 | P2 | P3 | P2 | P2 | P4 | P4 | P5 | P5 | P5 | P1 | P1 | P1 | P1 |
Average Waiting Time: 3.00 ms
Average Turnaround Time: 5.80 ms
```

```
4. Priority Scheduling
5. Shortest Remaining Time First
6. Longest remaining time first
7. Exit
Enter choice: 1

1. First-Come First-Served
2. Shortest Job First
3. Round Robin Scheduling
4. Priority Scheduling
5. Shortest Remaining Time First
6. Longest remaining time first
7. Exit
Enter choice: 2
```

Round Robin

OS_innovative.c output.txt

final > output.txt

```
Gantt Chart:
| P1 | P2 | P3 | P4 | P5 | P1 | P2 | P5 | P1 |
Average Waiting Time: 5.40 ms
Average Turnaround Time: 8.20 ms
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
2. Shortest Job First
3. Round Robin Scheduling
4. Priority Scheduling
5. Shortest Remaining Time First
6. Longest remaining time first
7. Exit
Enter choice: 3
Number of processes: 5
Process 1 Arrival Time: 0 Burst Time: 5
Process 2 Arrival Time: 1 Burst Time: 3
Process 3 Arrival Time: 2 Burst Time: 1
Process 4 Arrival Time: 3 Burst Time: 2
Process 5 Arrival Time: 4 Burst Time: 3
Enter time quantum: 2
```

Priority Scheduling

OS_innovative.c output.txt

final > output.txt

```
Gantt Chart:
0 | P1 | 4 | P2 | 11 | P3 | 14 | P4 | 16 | P5 | 20
Average Waiting Time: 3.20 ms
Average Turnaround Time: 7.20 ms
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Process 2 Arrival Time: 1 Burst Time: 3
Process 3 Arrival Time: 2 Burst Time: 1
Process 4 Arrival Time: 3 Burst Time: 2
Process 5 Arrival Time: 4 Burst Time: 3
Enter time quantum: 2

1. First-Come First-Served
2. Shortest Job First
3. Round Robin Scheduling
4. Priority Scheduling
5. Shortest Remaining Time First
6. Longest remaining time first
7. Exit
Enter choice: 4
```

SRTF

OS_innovative.c output.txt

final > output.txt

```
Gantt Chart:
0 | P3 | 3 | P2 | 5 | P4 | 7 | P5 | 10 | P1 | 14
Average Waiting Time: 3.00 ms
Average Turnaround Time: 5.80 ms
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1. First-Come First-Served
2. Shortest Job First
3. Round Robin Scheduling
4. Priority Scheduling
5. Shortest Remaining Time First
6. Longest remaining time first
7. Exit
Enter choice: 5
```

LRTF

OS_innovative.c output.txt

final > output.txt

```
Gantt Chart:
0 | P1 | 10 | P2 | 11 | P3 | 12 | P4 | 13 | P5 | 14
Average Waiting Time: 7.20 ms
Average Turnaround Time: 10.00 ms
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1. First-Come First-Served
2. Shortest Job First
3. Round Robin Scheduling
4. Priority Scheduling
5. Shortest Remaining Time First
6. Longest remaining time first
7. Exit
Enter choice: 6
```