Assignment 7
Operating Systems Lab
Arnav Samal | 122CS0107

Q1. Write a program to implement the First Come First Serve algorithm considering the arrival time of the process?
The algorithm should calculate the average waiting time, average turn around time

Code:

```
#include <stdio.h>


void waiting_turnaround_time(int wait_time[], int turn_around_time[], int n) {
        float total_wait_time = 0, total_tat = 0;
    for (int i = 0; i < n; i++) {
        total_wait_time += wait_time[i];
        total_tat += turn_around_time[i];
    }

    printf("Average waiting time: %.2f\n", total_wait_time / n);
    printf("Average turn around time: %.2f\n", total_tat / n);
}


void avg_time(int processes[], int n, int burst_time[], int arrival_time[]) {
    int wait_time[n], turn_around_time[n];
    wait_time[0] = 0;

    for (int i = 1; i < n; i++)
        wait_time[i] = burst_time[i - 1] + wait_time[i - 1] - arrival_time[i] +
arrival_time[i - 1];

    for (int i = 0; i < n; i++)
        turn_around_time[i] = burst_time[i] + wait_time[i];

    waiting_turnaround_time(wait_time, turn_around_time, n);
}


int main() {
    int processes[] = {0, 1, 2};
    int burst_time[] = {10, 5, 8};
    int arrival_time[] = {0, 1, 2};
    int n = sizeof(processes) / sizeof(processes[0]);
    avg_time(processes, n, burst_time, arrival_time);
    return 0;
```

}

Output:

```
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_7$ gcc q1.c
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_7$ ./a.out
Average waiting time: 7.33
Average turn around time: 15.00
```

Q2. Write a program to implement the Shortest Job First scheduling algorithm considering the preemption of the process and arrival time of the process? The algorithm should calculate the average waiting time, average turn around time

Code:

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


void waiting_turnaround_time(int wait_time[], int turn_around_time[], int n) {
        float total_wait_time = 0, total_tat = 0;
    for (int i = 0; i < n; i++) {
        total_wait_time += wait_time[i];
        total_tat += turn_around_time[i];
    }

    printf("Average waiting time: %.2f\n", total_wait_time / n);
    printf("Average turn around time: %.2f\n", total_tat / n);
}



void avg_time(int processes[], int n, int burst_time[], int arrival_time[]) {
    int remaining_time[n], wait_time[n], turn_around_time[n];
    for (int i = 0; i < n; i++)
        remaining_time[i] = burst_time[i];

    int complete = 0, t = 0, min_index, min_burst = 999;
    while (complete < n) {
        for (int j = 0; j < n; j++) {
            if (arrival_time[j] <= t && remaining_time[j] < min_burst &&
remaining_time[j] > 0) {
```

```
            min_burst = remaining_time[j];
            min_index = j;
        }
    }

    if (min_burst == 999) {
        t++;
        continue;
    }

    remaining_time[min_index]--;
    min_burst = remaining_time[min_index] == 0 ? 999 :
remaining_time[min_index];
    if (remaining_time[min_index] == 0) {
        complete++;
        turn_around_time[min_index] = t + 1 - arrival_time[min_index];
        wait_time[min_index] = turn_around_time[min_index] -
burst_time[min_index];
    }
    t++;
  }

  waiting_turnaround_time(wait_time, turn_around_time, n);
}



int main() {
  int processes[] = {0, 1, 2};
  int burst_time[] = {8, 4, 9};
  int arrival_time[] = {0, 1, 2};
  int n = sizeof(processes) / sizeof(processes[0]);
  avg_time(processes, n, burst_time, arrival_time);
  return 0;
}
```

Output:

```
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_7$ gcc q2.c
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_7$ ./a.out
Average waiting time: 4.67
Average turn around time: 11.67
```

Q3. Write a program to implement the priority scheduling algorithm considering the arrival time and preemption?
The algorithm should calculate the average waiting time, average turn around time

Code:

```
#include <stdio.h>


void waiting_turnaround_time(int wait_time[], int turn_around_time[], int n) {
        float total_wait_time = 0, total_tat = 0;
    for (int i = 0; i < n; i++) {
        total_wait_time += wait_time[i];
        total_tat += turn_around_time[i];
    }

    printf("Average waiting time: %.2f\n", total_wait_time / n);
    printf("Average turn around time: %.2f\n", total_tat / n);
}



void avg_time(int processes[], int n, int burst_time[], int arrival_time[], int priority[]) {
    int remaining_time[n], wait_time[n], turn_around_time[n];
    for (int i = 0; i < n; i++)
        remaining_time[i] = burst_time[i];

    int complete = 0, t = 0, min_index, min_priority;
    while (complete < n) {
        min_priority = 999;
        for (int j = 0; j < n; j++) {
            if (arrival_time[j] <= t && remaining_time[j] > 0 && priority[j] < min_priority) {
                min_priority = priority[j];
                min_index = j;
```

```
        }
    }

    if (min_priority == 999) {
        t++;
        continue;
    }

    remaining_time[min_index]--;
    if (remaining_time[min_index] == 0) {
        complete++;
        turn_around_time[min_index] = t + 1 - arrival_time[min_index];
        wait_time[min_index] = turn_around_time[min_index] -
burst_time[min_index];
    }
    t++;
  }

  waiting_turnaround_time(wait_time, turn_around_time, n);
}



int main() {
  int processes[] = {0, 1, 2};
  int burst_time[] = {10, 5, 8};
  int arrival_time[] = {0, 1, 2};
  int priority[] = {1, 2, 3};
  int n = sizeof(processes) / sizeof(processes[0]);
  avg_time(processes, n, burst_time, arrival_time, priority);
  return 0;
}
```

Output:

```
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_7$ gcc q3.c
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_7$ ./a.out
Average waiting time: 7.33
Average turn around time: 15.00
```

Q4. Write a program to implement the Round Robin scheduling algorithm considering the arrival time and preemption?

The algorithm should calculate the average waiting time, average turn around time

Code:

```
#include <stdio.h>



void waiting_turnaround_time(int wait_time[], int turn_around_time[], int n) {
        float total_wait_time = 0, total_tat = 0;
    for (int i = 0; i < n; i++) {
        total_wait_time += wait_time[i];
        total_tat += turn_around_time[i];
    }

    printf("Average waiting time: %.2f\n", total_wait_time / n);
    printf("Average turn around time: %.2f\n", total_tat / n);
}



void avg_time(int processes[], int n, int burst_time[], int arrival_time[], int quantum) {
    int remaining_time[n], wait_time[n], turn_around_time[n];
    for (int i = 0; i < n; i++)
        remaining_time[i] = burst_time[i];

    int t = 0, complete = 0;
    while (complete < n) {
        for (int i = 0; i < n; i++) {
            if (arrival_time[i] <= t && remaining_time[i] > 0) {
                if (remaining_time[i] > quantum) {
                    t += quantum;
                    remaining_time[i] -= quantum;
                } else {
                    t += remaining_time[i];
```

```
                wait_time[i] = t - burst_time[i] - arrival_time[i];
                turn_around_time[i] = t - arrival_time[i];
                remaining_time[i] = 0;
                complete++;
            }
        }
    }
  }

  waiting_turnaround_time(wait_time, turn_around_time, n);
}



int main() {
    int processes[] = {0, 1, 2};
    int burst_time[] = {10, 5, 8};
    int arrival_time[] = {0, 1, 2};
    int quantum = 3;
    int n = sizeof(processes) / sizeof(processes[0]);
    avg_time(processes, n, burst_time, arrival_time, quantum);
    return 0;
}
```

Output:

```
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_7$ gcc q4.c
nitr@nitr-HP-Compaq-Elite-8300-SFF:~/Downloads/122CS0107/OS/Lab_7$ ./a.out
Average waiting time: 11.00
Average turn around time: 18.67
```