Assignment Number 02

Name: Mihir Unmesh Patil

Roll NO: TYCOC213

Batch: C/C-3

CODE:

```
#include <iostream>
#include <cstdlib>
#include <queue>
#include <cstdio>
#include <algorithm>
using namespace std;
typedef struct process {
   int id, at, bt, remaining_bt, completion_time, pr;
   float wt, tat;
} process;
process p[10], p1[10], temp;
queue<int> q1;
int accept(int ch);
void turnwait(int n);
void display(int n);
void gantt_fcfs(int n);
void sjf_non_preemptive(int n);
void priority_non_preemptive(int n);
void priority_preemptive(int n, int* gantt_chart);
void ganttrr(int n);
void fcfs(int n);
void gantt_sjf(int n);
void sjf_preemptive(int n);
void gantt_priority_preemptive(int n, int* gantt_chart);
int main() {
int n, ch;
cout << "Choose Scheduling Algorithm: \n";
cout << "1. FCFS\n";
cout << "2. SJF (Non-Preemptive)\n";
cout << "3. SJF (Preemptive - SRTF)\n";
```

```
cout << "4. Preemptive Priority Scheduling\n";
cout << "5. Non-Preemptive Priority Scheduling\n";
cout << "6. Round Robin\n";
cin >> ch;
n = accept(ch);
switch (ch) {
case 1:
fcfs(n);
break;
case 2:
sjf_non_preemptive(n);
break;
case 3:
sjf_preemptive(n);
break;
case 4: {
int gantt_chart[100] = {0};
priority_preemptive(n, gantt_chart);
turnwait(n);
display(n);
gantt_priority_preemptive(n, gantt_chart);
break;
}
case 5:
priority_non_preemptive(n);
break;
case 6:
ganttrr(n);
break;
default:
cout << "Invalid choice!" << endl;
exit(1);
}
if (ch != 4) {
turnwait(n);
display(n);
```

```
}
return 0;
}
int accept(int ch) {
    int i, n;
    cout << "Enter the Total Number of Processes: ";
    cin >> n;
    if (n == 0) {
        cout << "Invalid number of processes!" << endl;
        exit(1);
    }
    cout << endl;
    for (i = 1; i <= n; i++) {
        cout << "Enter the Arrival Time of Process P" << i << ": ";
        cin >> p[i].at;
        p[i].id = i;
    }
    cout << endl;
    for (i = 1; i <= n; i++) {
        cout << "Enter the Burst Time of Process P" << i << ": ";
        cin >> p[i].bt;
        p[i].remaining_bt = p[i].bt;
    }
    if (ch == 3 || ch == 4) {
        for (i = 1; i <= n; i++) {
            cout << "Enter the Priority of Process P" << i << ": ";
            cin >> p[i].pr;
        }
    }
    for (i = 1; i <= n; i++) {
        p1[i] = p[i];
    }
    return n;
}
void turnwait(int n) {
    int i;
```

```
    for (i = 1; i <= n; i++) {
        p[i].tat = p[i].completion_time - p[i].at;
        p[i].wt = p[i].tat - p[i].bt;
        p[0].tat = p[0].tat + p[i].tat;
        p[0].wt = p[0].wt + p[i].wt;
    }
    p[0].tat = p[0].tat / n;
    p[0].wt = p[0].wt / n;
}
void display(int n) {
    int i;
    cout << "\n============================================================\n";
    cout << "\n\nHere AT = Arrival Time\nBT = Burst Time\nCT= Completion Time\nTAT = Turn Around Time\nWT =
Waiting Time\n";
    cout << "\n====================TABLE================================\n";
    cout << "Process\tAT\tBT\tCT\tTAT\t\tWT\n";
    for (i = 1; i <= n; i++) {
        printf("P%d\t%d\t%d\t%d\t%f\t%f\n", p[i].id, p[i].at, p[i].bt, p[i].completion_time, p[i].tat, p[i].wt);
    }
    cout << "\n============================================================\n";
    printf("\nAverage Turn Around Time: %f", p[0].tat);
    printf("\nAverage Waiting Time: %f\n", p[0].wt);
}
void fcfs(int n) {
    int i, current_time = 0;

    // Sort processes based on arrival time
    sort(p + 1, p + n + 1, [](process a, process b) {
        return a.at < b.at;
    });

    for (i = 1; i <= n; i++) {
        // If the current time is less than the arrival time of the process,
        // update the current time to the arrival time
        if (current_time < p[i].at) {
            current_time = p[i].at;
        }
```

```
        // The completion time is the current time plus the burst time
        p[i].completion_time = current_time + p[i].bt;

        // Update the current time to the completion time
        current_time = p[i].completion_time;
    }
    gantt_fcfs(n);
}
void gantt_fcfs(int n) {
    cout << "\nGantt Chart for FCFS Scheduling\n";
    for (int i = 1; i <= n; i++) {
        cout << "P" << p[i].id << " ";
    }
    cout << endl;
}
void sjf_preemptive(int n) {
int completed = 0, current_time = 0;
int current_process = -1;
bool is_completed[10] = {false};
while (completed < n) {
int shortest_time = 9999, next_process = -1;
for (int i = 1; i <= n; i++) {
if (!is_completed[i] && p[i].at <= current_time
&& p[i].remaining_bt < shortest_time &&
p[i].remaining_bt > 0) {
shortest_time = p[i].remaining_bt;
next_process = i;
}
}
if (next_process != -1) {
if (current_process != next_process) {
current_process = next_process;
}
p[current_process].remaining_bt--;
current_time++;
```

```
if (p[current_process].remaining_bt == 0) {
p[current_process].completion_time =
current_time;
is_completed[current_process] = true;
completed++;
current_process = -1;
}
} else {
current_time++;
}
}
}
void sjf_non_preemptive(int n) {
    int completed = 0, current_time = 0, smallest;
    bool is_completed[10] = {false};
    while (completed < n) {
        smallest = -1;
        int min_burst_time = 9999;
        for (int i = 1; i <= n; i++) {
            if (!is_completed[i] && p[i].at <= current_time && p[i].bt < min_burst_time) {
                min_burst_time = p[i].bt;
                smallest = i;
            }
        }
        if (smallest != -1) {
            current_time += p[smallest].bt;
            p[smallest].completion_time = current_time;
            is_completed[smallest] = true;
            completed++;
        } else {
            current_time++;
        }
    }
    gantt_sjf(n);
}
void gantt_sjf(int n) {
```

```
    cout << "\nGantt Chart for SJF Scheduling\n";
    for (int i = 1; i <= n; i++) {
        cout << "P" << p[i].id << " ";
    }
    cout << endl;
}
void priority_non_preemptive(int n) {
    int completed = 0, current_time = 0, smallest;
    bool is_completed[10] = {false};
    while (completed < n) {
        smallest = -1;
        int min_priority = 9999;
        for (int i = 1; i <= n; i++) {
            if (!is_completed[i] && p[i].at <= current_time && p[i].pr < min_priority) {
                min_priority = p[i].pr;
                smallest = i;
            }
        }
        if (smallest != -1) {
            current_time += p[smallest].bt;
            p[smallest].completion_time = current_time;
            is_completed[smallest] = true;
            completed++;
        } else {
            current_time++;
        }
    }
}
void priority_preemptive(int n, int* gantt_chart) {
    int completed = 0, current_time = 0;
    bool is_completed[10] = {false};
    int current_process = -1;
    while (completed < n) {
        int highest_priority = 9999, next_process = -1;
        for (int i = 1; i <= n; i++) {
            if (!is_completed[i] && p[i].at <= current_time && p[i].pr < highest_priority && p[i].remaining_bt > 0) {
```

```
                highest_priority = p[i].pr;
                next_process = i;
            }
        }
        if (next_process != -1) {
            if (current_process != next_process) {
                current_process = next_process;
            }
            p[current_process].remaining_bt--;
            gantt_chart[current_time] = current_process;
            current_time++;
            if (p[current_process].remaining_bt == 0) {
                p[current_process].completion_time = current_time;
                is_completed[current_process] = true;
                completed++;
                current_process = -1;
            }
        } else {
            gantt_chart[current_time] = 0;
            current_time++;
        }
    }
}
void gantt_priority_preemptive(int n, int* gantt_chart) {
    cout << "\nGantt Chart for Preemptive Priority Scheduling\n";
    for (int i = 0; i < 20; i++) {
        if (gantt_chart[i] != 0) {
            cout << "P" << gantt_chart[i] << " ";
        } else {
            cout << "Idle ";
        }
    }
    cout << endl;
}
void ganttrr(int n) {
    int i, ts, m, nextval, nextarr;
```

```
    nextval = p1[1].at;
    i = 1;
    cout << "\nEnter the Time Slice or Quantum: ";
    cin >> ts;
    for (i = 1; i <= n && p1[i].at <= nextval; i++) {
        q1.push(p1[i].id);
    }
    while (!q1.empty()) {
        m = q1.front();
        q1.pop();
        if (p1[m].bt >= ts) {
            nextval = nextval + ts;
        } else {
            nextval = nextval + p1[m].bt;
        }
        if (p1[m].bt >= ts) {
            p1[m].bt = p1[m].bt - ts;
        } else {
            p1[m].bt = 0;
        }
        while (i <= n && p1[i].at <= nextval) {
            q1.push(p1[i].id);
            i++;
        }
        if (p1[m].bt > 0) {
            q1.push(m);
        }
        if (p1[m].bt <= 0) {
            p[m].completion_time = nextval;
        }
    }
}
```

OUTPUT:

```
PS D:\Sem_6\OSL> cd "d:\Sem_6\OSL\" ; if ($?) { g++ Assignment_02_OSL.cpp
Choose Scheduling Algorithm:
1. FCFS
2. SJF (Non-Preemptive)
3. Preemptive Priority Scheduling
4. Non-Preemptive Priority Scheduling
5. Round Robin
1
Enter the Total Number of Processes: 6

Enter the Arrival Time of Process P1: 12
Enter the Arrival Time of Process P2: 3
Enter the Arrival Time of Process P3: 2
Enter the Arrival Time of Process P4: 3
Enter the Arrival Time of Process P5: 9
Enter the Arrival Time of Process P6: 15

Enter the Burst Time of Process P1: 7
Enter the Burst Time of Process P2: 3
Enter the Burst Time of Process P3: 1
Enter the Burst Time of Process P4: 2
Enter the Burst Time of Process P5: 6
Enter the Burst Time of Process P6: 8

Gantt Chart for FCFS Scheduling
P3 P2 P4 P5 P1 P6

==================================================


Here AT = Arrival Time
BT = Burst Time
CT= Completion Time
TAT = Turn Around Time
WT = Waiting Time

==================TABLE============================
Process AT      BT      CT      TAT             WT
P3      2       1       3       1.000000        0.000000
P2      3       3       6       3.000000        0.000000
P4      3       2       8       5.000000        3.000000
P5      9       6       15      6.000000        0.000000
P1      12      7       22      10.000000       3.000000
P6      15      8       30      15.000000       7.000000

==================================================

Average Turn Around Time: 6.666667
Average Waiting Time: 2.166667
```

```
PS D:\Sem_6\OSL> cd "d:\Sem_6\OSL\" ; if ($?) { g++ Assignment_02_OSL.cpp
Choose Scheduling Algorithm:
1. FCFS
2. SJF (Non-Preemptive)
3. Preemptive Priority Scheduling
4. Non-Preemptive Priority Scheduling
5. Round Robin
2
Enter the Total Number of Processes: 5

Enter the Arrival Time of Process P1: 2
Enter the Arrival Time of Process P2: 5
Enter the Arrival Time of Process P3: 1
Enter the Arrival Time of Process P4: 0
Enter the Arrival Time of Process P5: 4

Enter the Burst Time of Process P1: 6
Enter the Burst Time of Process P2: 2
Enter the Burst Time of Process P3: 8
Enter the Burst Time of Process P4: 3
Enter the Burst Time of Process P5: 4

Gantt Chart for SJF Scheduling
P1 P2 P3 P4 P5

==================================================


Here AT = Arrival Time
BT = Burst Time
CT= Completion Time
TAT = Turn Around Time
WT = Waiting Time

==================TABLE============================
Process AT      BT      CT      TAT             WT
P1      2       6       9       7.000000        1.000000
P2      5       2       11      6.000000        4.000000
P3      1       8       23      22.000000       14.000000
P4      0       3       3       3.000000        0.000000
P5      4       4       15      11.000000       7.000000

==================================================

Average Turn Around Time: 9.800000
Average Waiting Time: 5.200000
```

```
1. FCFS
2. SJF (Non-Preemptive)
3. Preemptive Priority Scheduling
4. Non-Preemptive Priority Scheduling
5. Round Robin
3
Enter the Total Number of Processes: 5

Enter the Arrival Time of Process P1: 0
Enter the Arrival Time of Process P2: 5
Enter the Arrival Time of Process P3: 12
Enter the Arrival Time of Process P4: 2
Enter the Arrival Time of Process P5: 9

Enter the Burst Time of Process P1: 11
Enter the Burst Time of Process P2: 28
Enter the Burst Time of Process P3: 2
Enter the Burst Time of Process P4: 10
Enter the Burst Time of Process P5: 16
Enter the Priority of Process P1: 2
Enter the Priority of Process P2: 0
Enter the Priority of Process P3: 3
Enter the Priority of Process P4: 1
Enter the Priority of Process P5: 4

==================================================


Here AT = Arrival Time
BT = Burst Time
CT= Completion Time
TAT = Turn Around Time
WT = Waiting Time

==================TABLE============================
Process AT      BT      CT      TAT             WT
P1      0       11      49      49.000000       38.000000
P2      5       28      33      28.000000       0.000000
P3      12      2       51      39.000000       37.000000
P4      2       10      40      38.000000       28.000000
P5      9       16      67      58.000000       42.000000

==================================================

Average Turn Around Time: 42.400002
Average Waiting Time: 29.000000

Gantt Chart for Preemptive Priority Scheduling
P1 P1 P4 P4 P4 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2 P2
```

```
Choose Scheduling Algorithm:
1. FCFS
2. SJF (Non-Preemptive)
3. Preemptive Priority Scheduling
4. Non-Preemptive Priority Scheduling
5. Round Robin
4
Enter the Total Number of Processes: 4

Enter the Arrival Time of Process P1: 0
Enter the Arrival Time of Process P2: 2
Enter the Arrival Time of Process P3: 3
Enter the Arrival Time of Process P4: 5

Enter the Burst Time of Process P1: 10
Enter the Burst Time of Process P2: 5
Enter the Burst Time of Process P3: 2
Enter the Burst Time of Process P4: 20
Enter the Priority of Process P1: 2
Enter the Priority of Process P2: 1
Enter the Priority of Process P3: 0
Enter the Priority of Process P4: 3

==================================================


Here AT = Arrival Time
BT = Burst Time
CT= Completion Time
TAT = Turn Around Time
WT = Waiting Time

==================TABLE============================
Process AT      BT      CT      TAT             WT
P1      0       10      10      10.000000       0.000000
P2      2       5       17      15.000000       10.000000
P3      3       2       12      9.000000        7.000000
P4      5       20      37      32.000000       12.000000

==================================================

Average Turn Around Time: 16.500000
Average Waiting Time: 7.250000
```

```
Choose Scheduling Algorithm:
1. FCFS
2. SJF (Non-Preemptive)
3. SJF (Preemptive - SRTF)
4. Preemptive Priority Scheduling
5. Non-Preemptive Priority Scheduling
6. Round Robin
3
Enter the Total Number of Processes: 3

Enter the Arrival Time of Process P1: 0
Enter the Arrival Time of Process P2: 1
Enter the Arrival Time of Process P3: 2

Enter the Burst Time of Process P1: 6
Enter the Burst Time of Process P2: 3
Enter the Burst Time of Process P3: 7
Enter the Priority of Process P1: 0
Enter the Priority of Process P2: 0
Enter the Priority of Process P3: 0

==================================================


Here AT = Arrival Time
BT = Burst Time
CT= Completion Time
TAT = Turn Around Time
WT = Waiting Time

==================TABLE============================
Process AT      BT      CT      TAT             WT
P1      0       6       9       9.000000        3.000000
P2      1       3       4       3.000000        0.000000
P3      2       7       16      14.000000       7.000000

==================================================

Average Turn Around Time: 8.666667
Average Waiting Time: 3.333333
```

```
Choose Scheduling Algorithm:
1. FCFS
2. SJF (Non-Preemptive)
3. Preemptive Priority Scheduling
4. Non-Preemptive Priority Scheduling
5. Round Robin
5
Enter the Total Number of Processes: 6

Enter the Arrival Time of Process P1: 0
Enter the Arrival Time of Process P2: 1
Enter the Arrival Time of Process P3: 2
Enter the Arrival Time of Process P4: 3
Enter the Arrival Time of Process P5: 4
Enter the Arrival Time of Process P6: 6

Enter the Burst Time of Process P1: 4
Enter the Burst Time of Process P2: 5
Enter the Burst Time of Process P3: 2
Enter the Burst Time of Process P4: 1
Enter the Burst Time of Process P5: 6
Enter the Burst Time of Process P6: 3

Enter the Time Slice or Quantum: 2

==================================================


Here AT = Arrival Time
BT = Burst Time
CT= Completion Time
TAT = Turn Around Time
WT = Waiting Time

==================TABLE============================
Process AT      BT      CT      TAT             WT
P1      0       4       8       8.000000        4.000000
P2      1       5       18      17.000000       12.000000
P3      2       2       6       4.000000        2.000000
P4      3       1       9       6.000000        5.000000
P5      4       6       21      17.000000       11.000000
P6      6       3       19      13.000000       10.000000

==================================================

Average Turn Around Time: 10.833333
Average Waiting Time: 7.333333
```