

Rajneesh Pandey - 106119100

Question 1:**Program for Shortest Job First (or SJF) CPU Scheduling | (Non- preemptive)**

Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next. SJN is a non-preemptive algorithm.

Program and Input/Output

```

1 //106119100 Rajneesh Pandey
2 #include <iostream>
3 using namespace std;
4 int mat[10][6];
5 void swap(int *a, int *b){
6     int temp = *a;
7     *a = *b;
8     *b = temp;}
9 void arrangeArrival(int num, int mat[][6])
10 { for (int i = 0; i < num; i++){
11     for (int j = 0; j < num - i - 1; j++){
12         if (mat[j][1] > mat[j + 1][1]){
13             for (int k = 0; k < 5; k++){
14                 swap(mat[j][k], mat[j + 1][k]);
15             }
16         }
17     }
18 }
19 void completionTime(int num, int mat[][6]){
20     int temp, val;
21     mat[0][3] = mat[0][1] + mat[0][2];
22     mat[0][5] = mat[0][3] - mat[0][1];
23     mat[0][4] = mat[0][5] - mat[0][2];
24     for (int i = 1; i < num; i++){
25         temp = mat[i - 1][3];
26         int low = mat[i][2];
27         for (int j = i; j < num; j++){
28             if (temp >= mat[j][1] && low >= mat[j][2]){
29                 low = mat[j][2]; val = j;
30             }
31         }
32         mat[val][3] = temp + mat[val][2];
33         mat[val][5] = mat[val][3] - mat[val][1];
34         mat[val][4] = mat[val][5] - mat[val][2];
35         for (int k = 0; k < 6; k++){
36             swap(mat[val][k], mat[i][k]);
37         }
38     }
39 }
40 int main()
41 {
42     int num, temp;
43     cout << "Enter number of Process: ";
44     cin >> num;
45     cout << "...Enter the process ID...\n";
46     for (int i = 0; i < num; i++){
47         { cout << "...Process " << i + 1 << "... \n";
48             cout << "Enter Process Id: "; cin >> mat[i][0];
49             cout << "Enter Arrival Time: "; cin >> mat[i][1];
50             cout << "Enter Burst Time: "; cin >> mat[i][2];
51             cout << "Before Arrange...\n"; cout << "Process ID\tArrival Time\tBurst Time\n";
52             for (int i = 0; i < num; i++){
53                 cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t" << mat[i][2] << "\n";
54             }
55             arrangeArrival(num, mat);
56             completionTime(num, mat);
57             cout << "Final Result...\n";
58             cout << "Process ID\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n";
59             for (int i = 0; i < num; i++){
60                 { cout << mat[i][0] << "\t\t" << mat[i][1] << "\t\t" << mat[i][2] << "\t\t" << mat[i][4] <<
61                     "\t\t" << mat[i][5] << "\n";
62             }
63         }
64     }
65 }

```

```

rajneesh@rajneesh-VirtualBox: ~/Desktop/OS Lab/5$ g++ SJF_Non_Preemptive.cpp
-o sjfnp
rajneesh@rajneesh-VirtualBox: ~/Desktop/OS Lab/5$ ./sjfnp
Enter number of Process: 4
...Enter the process ID...
...Process 1...
Enter Process Id: 1
Enter Arrival Time: 2
Enter Burst Time: 3
...Process 2...
Enter Process Id: 2
Enter Arrival Time: 0
Enter Burst Time: 4
...Process 3...
Enter Process Id: 3
Enter Arrival Time: 4
Enter Burst Time: 2
...Process 4...
Enter Process Id: 4
Enter Arrival Time: 5
Enter Burst Time: 4
Before Arrange...
Process ID      Arrival Time    Burst Time
1               2               3
2               0               4
3               4               2
4               5               4
Final Result...
Process ID      Arrival Time    Burst Time      Waiting Time    Turnaround
Time
2               0               4               0               4
3               4               2               0               2
1               2               3               4               7
4               5               4               4               8
rajneesh@rajneesh-VirtualBox: ~/Desktop/OS Lab/5$

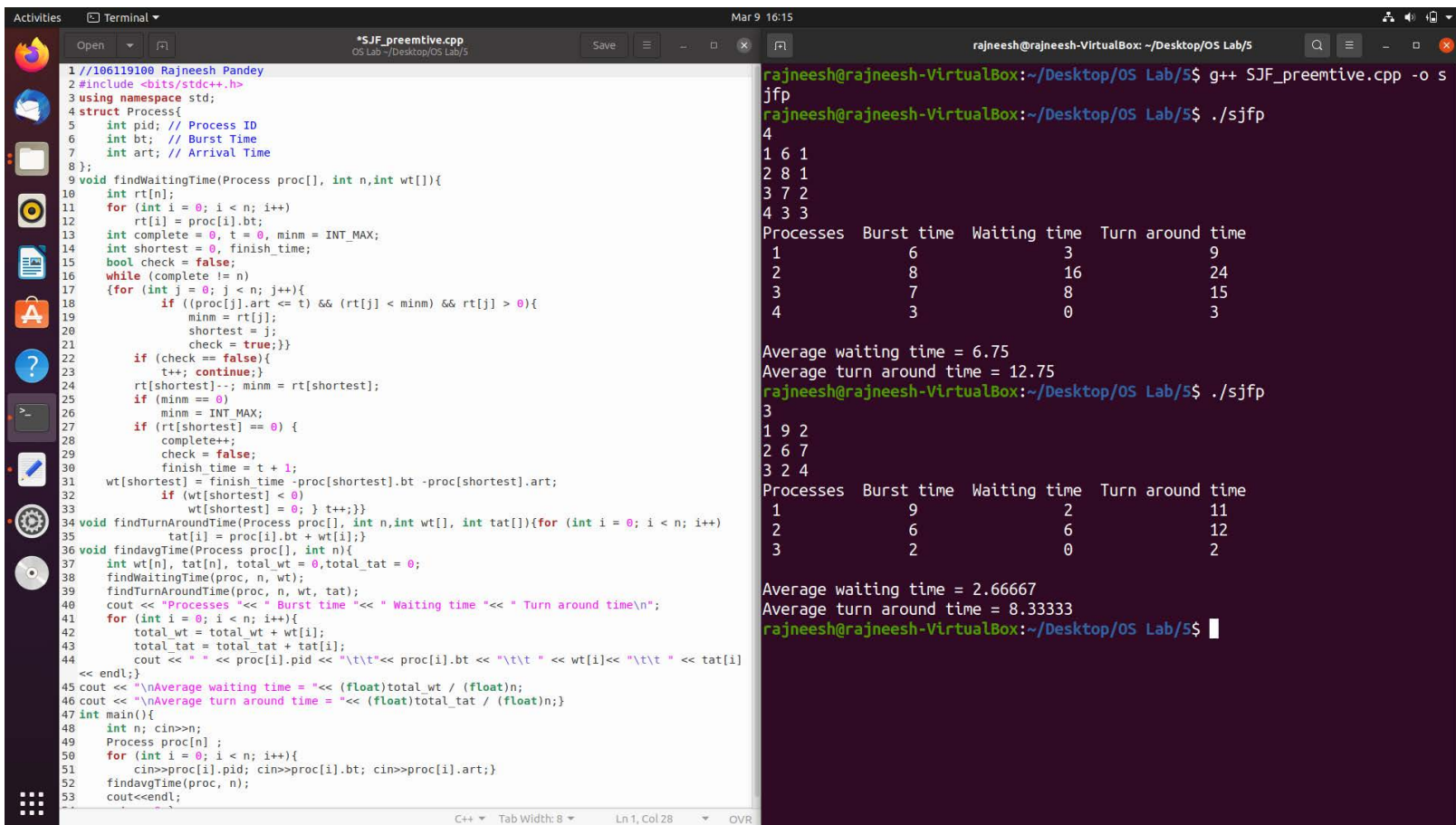
```

Question 2:

Program for Shortest Job First (SJF) scheduling | (Preemptive)

Shortest Remaining Time First (SRTF) scheduling

In the Shortest Remaining Time First (SRTF) scheduling algorithm, the process with the smallest amount of time remaining until completion is selected to execute. Since the currently executing process is the one with the shortest amount of time remaining by definition, and since that time should only reduce as execution progresses, processes will always run until they complete or a new process is added that requires a smaller amount of time.



```
1 //106119100 Rajneesh Pandey
2 #include <bits/stdc++.h>
3 using namespace std;
4 struct Process{
5     int pid; // Process ID
6     int bt; // Burst Time
7     int art; // Arrival Time
8 };
9 void findWaitingTime(Process proc[], int n,int wt[]){
10     int rt[n];
11     for (int i = 0; i < n; i++){
12         rt[i] = proc[i].bt;
13         int complete = 0, t = 0, minm = INT_MAX;
14         int shortest = 0, finish_time;
15         bool check = false;
16         while (complete != n)
17         {for (int j = 0; j < n; j++){
18             if ((proc[j].art <= t) && (rt[j] < minm) && rt[j] > 0){
19                 minm = rt[j];
20                 shortest = j;
21                 check = true;}}
22             if (check == false){
23                 t++; continue;}
24             rt[shortest]--; minm = rt[shortest];
25             if (minm == 0)
26                 minm = INT_MAX;
27             if (rt[shortest] == 0) {
28                 complete++;
29                 check = false;
30                 finish_time = t + 1;
31                 wt[shortest] = finish_time - proc[shortest].bt - proc[shortest].art;
32                 if (wt[shortest] < 0)
33                     wt[shortest] = 0; } t++;}}
34 void findTurnAroundTime(Process proc[], int n,int wt[], int tat[]){for (int i = 0; i < n; i++){
35     tat[i] = proc[i].bt + wt[i];}
36 void findavgTime(Process proc[], int n){
37     int wt[n], tat[n], total_wt = 0, total_tat = 0;
38     findWaitingTime(proc, n, wt);
39     findTurnAroundTime(proc, n, wt, tat);
40     cout << "Processes " << " Burst time " << " Waiting time " << " Turn around time\n";
41     for (int i = 0; i < n; i++){
42         total_wt = total_wt + wt[i];
43         total_tat = total_tat + tat[i];
44         cout << " " << proc[i].pid << "\t\t" << proc[i].bt << "\t\t" << wt[i] << "\t\t" << tat[i]
45         << endl;}
46 cout << "\nAverage waiting time = "<< (float)total_wt / (float)n;
47 cout << "\nAverage turn around time = "<< (float)total_tat / (float)n;
48 int main(){
49     int n; cin>>n;
50     Process proc[n] ;
51     for (int i = 0; i < n; i++){
52         cin>>proc[i].pid; cin>>proc[i].bt; cin>>proc[i].art;}
53     findavgTime(proc, n);
54     cout<<endl;
```

```
rajneesh@rajneesh-VirtualBox: ~/Desktop/OS Lab/5$ g++ SJF_preemptive.cpp -o sjfp
rajneesh@rajneesh-VirtualBox:~/Desktop/OS Lab/5$ ./sjfp
4
1 6 1
2 8 1
3 7 2
4 3 3
Processes Burst time Waiting time Turn around time
1 6 3 9
2 8 16 24
3 7 8 15
4 3 0 3
Average waiting time = 6.75
Average turn around time = 12.75
rajneesh@rajneesh-VirtualBox:~/Desktop/OS Lab/5$ ./sjfp
3
1 9 2
2 6 7
3 2 4
Processes Burst time Waiting time Turn around time
1 9 2 11
2 6 6 12
3 2 0 2
Average waiting time = 2.66667
Average turn around time = 8.33333
rajneesh@rajneesh-VirtualBox:~/Desktop/OS Lab/5$
```

Question 2:

Program for Priority CPU Scheduling |

Priority scheduling is one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with the highest priority is to be executed first and so on.

Processes with the same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

```
1 //106119100 Rajneesh Pandey
2 #include <bits/stdc++.h>
3 using namespace std;
4 struct Process{
5     int pid; // Process ID
6     int bt; // CPU Burst time required
7     int priority; // Priority of this process
8 };
9 bool comparison(Process a, Process b){
10     return (a.priority > b.priority);
11 }
12 void findWaitingTime(Process proc[], int n, int wt[]){
13     wt[0] = 0;
14     for (int i = 1; i < n; i++){
15         wt[i] = proc[i-1].bt + wt[i-1];
16     }
17 void findTurnAroundTime(Process proc[], int n, int wt[], int tat[]){
18     for (int i = 0; i < n; i++){
19         tat[i] = proc[i].bt + wt[i];
20     }
21 void findavgTime(Process proc[], int n){
22     int wt[n], tat[n], total_wt = 0, total_tat = 0;
23     findWaitingTime(proc, n, wt);
24     findTurnAroundTime(proc, n, wt, tat);
25
26     cout << "\nProcesses " << "Burst time " << "Waiting time " << "Turn around time\n";
27     for (int i = 0; i < n; i++){
28         total_wt = total_wt + wt[i];
29         total_tat = total_tat + tat[i];
30         cout << " " << proc[i].pid << " \t\t" << proc[i].bt << " \t\t " << wt[i] << " \t\t " << tat[i] << endl;
31     }
32     cout << "\nAverage waiting time = " << (float)total_wt / (float)n;
33     cout << "\nAverage turn around time = " << (float)total_tat / (float)n;
34 }
35 void priorityScheduling(Process proc[], int n){
36     sort(proc, proc + n, comparison);
37     cout << "Order in which processes gets executed \n";
38     for (int i = 0; i < n; i++){
39         cout << proc[i].pid << " ";
40     }
41     findavgTime(proc, n);
42 }
43 int main(){
44     int n;
45     cin >> n;
46     Process proc[n];
47     for (int i = 0; i < n; i++){
48         cin >> proc[i].pid;
49         cin >> proc[i].bt;
50         cin >> proc[i].priority;
51     }
52     priorityScheduling(proc, n);
53     cout << endl;
54     return 0;
55 }
```

```
rajneesh@rajneesh-VirtualBox: ~/Desktop/OS Lab/5$ g++ priority_scheduling.cpp
p -o ps
rajneesh@rajneesh-VirtualBox: ~/Desktop/OS Lab/5$ ./ps
3
1 10 2
2 5 0
3 8 1
Order in which processes gets executed
1 3 2
Processes Burst time Waiting time Turn around time
1 10 0 10
3 8 10 18
2 5 18 23
Average waiting time = 9.33333
Average turn around time = 17
rajneesh@rajneesh-VirtualBox: ~/Desktop/OS Lab/5$ ./ps
4
1 4 2
2 10 6
3 8 3
4 6 9
Order in which processes gets executed
4 2 3 1
Processes Burst time Waiting time Turn around time
4 6 0 6
2 10 6 16
3 8 16 24
1 4 24 28
Average waiting time = 11.5
Average turn around time = 18.5
rajneesh@rajneesh-VirtualBox: ~/Desktop/OS Lab/5$
```