Assignment 1

Operating System

Name: Etcherla Sai Manoj **Mis. No:** 112015044 **Branch:** CSE

1. Implementation of First Come First Serve Scheduling Algorithm

Case 1:

Input: Processes and their Burst time (Minimum No. of Process 4)
Output: Average waiting time, Average Turnaround Time

Code:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){

    // Input number of process
    int noOfProcess;
    cout << "Enter number of Process : ";
    cin >> noOfProcess;

    // Input burst time for every process
    int burstTime[noOfProcess];
    for(int i = 0; i < noOfProcess; i++){
        cout << "Enter Burst time for Process " << i << " : ";
        cin >> burstTime[i];
    }

    // Initializing arrays for waiting time and turn around time
    int waitingTime[noOfProcess], turnAroundTime[noOfProcess];
    waitingTime[0] = 0;
    turnAroundTime[0] = burstTime[0];

    // Initialize total waiting time and total turn around time variables
    float totalWT = waitingTime[0], totalTAT = turnAroundTime[0];

    // Calculating waiting and turn around time for every process
    for(int i = 1; i < noOfProcess; i++){
        waitingTime[i] = waitingTime[i-1] + burstTime[i-1];
        turnAroundTime[i] = waitingTime[i] + burstTime[i];
        totalWT += waitingTime[i];
        totalTAT += turnAroundTime[i];
    }

    // Output average waiting  time and average turn around time
    cout << "\nAverage Waiting time : " << totalWT / (float)noOfProcess;
    cout << "\nAverage Turn Around time : " << totalTAT / (float)noOfProcess;

    return 0;
}
```

Output:

```
PS C:\Users\DELL\OneDrive\Desktop\Labs> cd "c:\Users\DELL\OneDrive\Desktop\Labs\IIIT PUNE LABS\
4 Fourth Sem\OS LAB\LAB 1\" ; if ($?) { g++ FCFS1.cpp -o FCFS1 } ; if ($?) { .\FCFS1 }
Enter number of Process : 4
Enter Burst time for Process 0 : 21
Enter Burst time for Process 1 : 3
Enter Burst time for Process 2 : 6
Enter Burst time for Process 3 : 2

Average Waiting time : 18.75
Average Turn Around time : 26.75
PS C:\Users\DELL\OneDrive\Desktop\Labs\IIIT PUNE LABS\4 Fourth Sem\OS LAB\LAB 1>
```

Case 2:
Input: Processes, their Burst time, and Arrival time (Minimum No. of Process 4)
Output: Average waiting time, Average Turnaround Time

Code:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){

    // Input number of process
    int noOfProcess;
    cout << "Enter number of Process : ";
    cin >> noOfProcess;

    // Input burst time and arrival time for every process
    vector<vector<int>> x;
    for(int i = 0; i < noOfProcess; i++){
        vector<int> y;
        int burstTime, arrivalTime;
        cout << "Enter Burst time for Process " << i << " : ";
        cin >> burstTime;
        cout << "Enter Arrival time for Process " << i << " : ";
        cin >> arrivalTime;
        y.push_back(arrivalTime);
        y.push_back(burstTime);
        x.push_back(y);
    }
    // Sort processes based on arrival times
    sort(x.begin(), x.end());

    // Initializing arrays for waiting time, turn around time and finish time
    int waitingTime[noOfProcess], turnAroundTime[noOfProcess], finishTime[noOfProcess];

    // Initialize total waiting time and total turn around time variables
    float totalWT = 0, totalTAT = 0;

    // Calculating waiting time, turn around time and finish time for every process
    for(int i = 0; i < x.size(); i++){
        if(i == 0){
            finishTime[i] = x[i][0] + x[i][1];
            turnAroundTime[i] = finishTime[i] - x[i][0];
            waitingTime[i] = turnAroundTime[i] - x[i][1];
        }
        else{
            if(finishTime[i-1] > x[i][0]){
                finishTime[i] = x[i][1] + finishTime[i-1];
            }
            else{
                finishTime[i] = x[i][1] + x[i][0];
            }
            turnAroundTime[i] = finishTime[i] - x[i][0];
            waitingTime[i] = turnAroundTime[i] - x[i][1];
        }
        totalWT += waitingTime[i];
        totalTAT += turnAroundTime[i];
    }

    // Output average waiting  time and average turn around time
    cout << "\nAverage Waiting time : " << totalWT / (float)noOfProcess;
    cout << "\nAverage Turn Around time : " << totalTAT / (float)noOfProcess;

    return 0;
}
```

Output:

```
PS C:\Users\DELL\OneDrive\Desktop\Labs> cd "c:\Users\DELL\OneDrive\Desktop\Labs\IIIT PUNE LABS\
4 Fourth Sem\OS LAB\LAB 1\" ; if ($?) { g++ FCFS2.cpp -o FCFS2 } ; if ($?) { .\FCFS2 }
Enter number of Process : 5
Enter Burst time for Process 0 : 6
Enter Arrival time for Process 0 : 2
Enter Burst time for Process 1 : 2
Enter Arrival time for Process 1 : 5
Enter Burst time for Process 2 : 8
Enter Arrival time for Process 2 : 1
Enter Burst time for Process 3 : 3
Enter Arrival time for Process 3 : 0
Enter Burst time for Process 4 : 4
Enter Arrival time for Process 4 : 4

Average Waiting time : 8
Average Turn Around time : 12.6
PS C:\Users\DELL\OneDrive\Desktop\Labs\IIIT PUNE LABS\4 Fourth Sem\OS LAB\LAB 1>
```

2. Implementation of SJF Scheduling Algorithm.
(Minimum No. of Process 4)

Case 1: Non-Preemptive
Input: Processes, Burst time, and Arrival time are given as input (Minimum No. of Process 4)
Output: Average waiting time, Average Turnaround Time

Code:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){

    // Input number of process
    int noOfProcess;
    cout << "Enter number of Process : ";
    cin >> noOfProcess;

    // Input burst time for every process
    int burstTime[noOfProcess];
    for(int i = 0; i < noOfProcess; i++){
        cout << "Enter Burst time for Process " << i << " : ";
        cin >> burstTime[i];
    }

    // Sort processes based on burst times
    sort(burstTime, burstTime+noOfProcess);

    // Initializing arrays for waiting time and turn around time
    int waitingTime[noOfProcess], turnAroundTime[noOfProcess];
    waitingTime[0] = 0;
    turnAroundTime[0] = burstTime[0];

    // Initialize total waiting time and total turn around time variables
    float totalWT = waitingTime[0], totalTAT = turnAroundTime[0];

    // Calculating waiting and turn around time for every process
    for(int i = 1; i < noOfProcess; i++){
        waitingTime[i] = waitingTime[i-1] + burstTime[i-1];
        turnAroundTime[i] = waitingTime[i] + burstTime[i];
        totalWT = totalWT + waitingTime[i];
        totalTAT = totalTAT + turnAroundTime[i];
    }

    // Output average waiting  time and average turn around time
    cout << "\nAverage Waiting time : " << totalWT / (float)noOfProcess;
    cout << "\nAverage Turn Around time : " << totalTAT / (float)noOfProcess;

    return 0;
}
```

Output:

```
PS C:\Users\DELL\OneDrive\Desktop\Labs> cd "c:\Users\DELL\OneDrive\Desktop\Labs\IIIT PUNE LABS\
4 Fourth Sem\OS LAB\LAB 1\" ; if ($?) { g++ SJF__Non_Preemptive.cpp -o SJF__Non_Preemptive } ;
if ($?) { .\SJF__Non_Preemptive }
Enter number of Process : 4
Enter Burst time for Process 0 : 6
Enter Burst time for Process 1 : 8
Enter Burst time for Process 2 : 7
Enter Burst time for Process 3 : 3

Average Waiting time : 7
Average Turn Around time : 13
PS C:\Users\DELL\OneDrive\Desktop\Labs\IIIT PUNE LABS\4 Fourth Sem\OS LAB\LAB 1>
```

Case 2: **Preemptive.**
Input: Processes, Burst time, and Arrival time are given as input (Minimum No. of Process 4)
Output: Average waiting time, Average Turnaround Time

Code:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){

    // Input number of process
    int noOfProcess;
    cout << "Enter number of Process : ";
    cin >> noOfProcess;

    // Input burst time and arrival time for every process
    vector<vector<int>> x;
    for(int i = 0; i < noOfProcess; i++){
        vector<int> y;
        int burstTime, arrivalTime;
        cout << "Enter Burst time for Process " << i << " : ";
        cin >> burstTime;
        cout << "Enter Arrival time for Process " << i << " : ";
        cin >> arrivalTime;
        y.push_back(burstTime);
        y.push_back(arrivalTime);
        x.push_back(y);
    }

    // Initializing arrays for waiting time, turn around time and finish time
    int waitingTime[noOfProcess], turnAroundTime[noOfProcess];

    // Initialize total waiting time and total turn around time variables
    float totalWT = 0, totalTAT = 0;

    int burstTime[noOfProcess];
    for (int i = 0; i < noOfProcess; i++){
        burstTime[i] = x[i][0];
    }

    int finished = 0, time = 0, minimum = INT_MAX, shortest = 0, finishTime;
    bool check = false;

    // Until all processes gets completed
    while (finished != noOfProcess) {

        // Check process with minimum remaining time
        for (int j = 0; j < noOfProcess; j++) {
            if ((x[j][1] <= time) && (burstTime[j] < minimum) && burstTime[j] > 0){
                minimum = burstTime[j];
                shortest = j;
                check = true;
            }
        }
        if (check == false) {
            time++;
            continue;
        }
        // Decrement time by one
        burstTime[shortest]--;
        minimum = burstTime[shortest];
        if (minimum == 0){
            minimum = INT_MAX;
        }
        // Process is completely executed
        if (burstTime[shortest] == 0) {
            finished++;
            check = false;
            finishTime = time + 1;
            // Calculating Waiting time
            waitingTime[shortest] = finishTime - x[shortest][0] - x[shortest][1];
            if (waitingTime[shortest] < 0){
                waitingTime[shortest] = 0;
            }
        }
        // Increment time
        time++;
    }

    // Calculating Turn Around time for every process
    for (int i = 0; i < noOfProcess; i++){
        turnAroundTime[i] = x[i][0] + waitingTime[i];
        totalWT = totalWT + waitingTime[i];
        totalTAT = totalTAT + turnAroundTime[i];
    }

    // Output average waiting  time and average turn around time
    cout << "\nAverage Waiting time : " << totalWT / (float)noOfProcess;
    cout << "\nAverage Turn Around time : " << totalTAT / (float)noOfProcess;


    return 0;
}
```

Output:

```
PS C:\Users\DELL\OneDrive\Desktop\Labs> cd "c:\Users\DELL\OneDrive\Desktop\Labs\IIIT PUNE LABS\
4 Fourth Sem\OS LAB\LAB 1\" ; if ($?) { g++ SJF__Preemptive.cpp -o SJF__Preemptive } ; if ($?)
{ .\SJF__Preemptive }
Enter number of Process : 5
Enter Burst time for Process 0 : 6
Enter Arrival time for Process 0 : 2
Enter Burst time for Process 1 : 2
Enter Arrival time for Process 1 : 5
Enter Burst time for Process 2 : 8
Enter Arrival time for Process 2 : 1
Enter Burst time for Process 3 : 3
Enter Arrival time for Process 3 : 0
Enter Burst time for Process 4 : 4
Enter Arrival time for Process 4 : 4

Average Waiting time : 4.6
Average Turn Around time : 9.2
PS C:\Users\DELL\OneDrive\Desktop\Labs\IIIT PUNE LABS\4 Fourth Sem\OS LAB\LAB 1>
```