Lab No. 5 **Date: 2025/09/01**

Title: Write a program to calculate the average turnaround time and waiting time for user input process parameters using SJF process scheduling algorithm.

Introduction

SJF is a non-preemptive scheduling algorithm where the CPU is assigned to the process with the shortest burst time among the available processes in the ready queue. Once a process starts execution, it cannot be interrupted.

Characteristics:

- Non-preemptive: A running process cannot be stopped until it finishes.
- Optimal in terms of minimizing average waiting time if all burst times are known.
- May cause starvation for long processes if short processes keep arriving.

Algorithm:

1. Sort processes by arrival time.
2. At any time, select the available process with the shortest burst time.
3. Execute the selected process until completion.
4. Repeat step 2 until all processes are completed.

Programming Language: C++

IDE: Dev C++

Source Code:

```
#include <iostream>
#include <iomanip>
using namespace std;
struct Process
{
   string name;
   int arrival;
   int burst;
   int completion;
   int turnaround;
   int waiting;
   bool done;
};
int main()
{
   cout << "====================================" << endl;
   cout << "||    SJF Scheduling (NP)       ||" << endl;
   cout << "|| Compiled by :- Jonash Chataut  ||" << endl;
   cout << "====================================" << endl << endl;
   int n;
   cout << "Enter number of processes: ";
   cin >> n;
   Process p[n];
   cout << "Enter process name, arrival time, and burst time:\n";
   for (int i = 0; i < n; i++)
   {
      cin >> p[i].name >> p[i].arrival >> p[i].burst;
      p[i].done = false;
   }
   int currentTime = 0, completed = 0;
   double totalTAT = 0, totalWT = 0;
   // Print Gantt Chart
   cout << "\nGantt Chart:\n";
   cout << "------------------------------------------------\n";
   while (completed < n)
   {
      int idx = -1;
      int minBT = 1e9;
      // Find the shortest job among arrived processes
      for (int i = 0; i < n; i++)
      {
         if (!p[i].done && p[i].arrival <= currentTime)
         {
            if (p[i].burst < minBT)
            {
               minBT = p[i].burst;
               idx = i;
            }
```

```
        }
      }
      if (idx == -1)
      {
        currentTime++; // CPU idle
        continue;
      }
      cout << "|   " << p[idx].name << "   ";
      // Calculate times
      p[idx].completion = currentTime + p[idx].burst;
      currentTime = p[idx].completion;
      p[idx].turnaround = p[idx].completion - p[idx].arrival; // TAT = CT - AT
      p[idx].waiting = p[idx].turnaround - p[idx].burst;     // WT = TAT - BT
      totalTAT += p[idx].turnaround;
      totalWT += p[idx].waiting;
      p[idx].done = true;
      completed++;
    }
    cout << "|\n";
    cout << "-------------------------------------------------\n";
    // Print timeline
    currentTime = 0;
    cout << 0;
    for (int i = 0; i < n; i++)
    {
      if (p[i].done)
      {
        cout << setw(8) << p[i].completion;
      }
    }
    cout << "\n";
    // Print Process Table
    cout << "\nProcess\tAT\tBT\tCT\tTAT (CT-AT)\tWT (TAT-BT)\n";
    cout << "-----------------------------------------------------------------\n";
    for (int i = 0; i < n; i++)
    {
      cout << p[i].name << "\t"
           << p[i].arrival << "\t"
           << p[i].burst << "\t"
           << p[i].completion << "\t"
           << p[i].completion << " - " << p[i].arrival << " = " << p[i].turnaround << "\t"
           << p[i].turnaround << " - " << p[i].burst << " = " << p[i].waiting << "\n";
    }
    cout << "\nAverage Turnaround Time = " << fixed << setprecision(2) << (totalTAT / n);
    cout << "\nAverage Waiting Time = " << fixed << setprecision(2) << (totalWT / n) << "\n";
    return 0;
  }
```

Output:

```
C:\csit\fourth_sem_jonash\OS

====================================
||      SJF Scheduling (NP)        ||
|| Compiled by :- Jonash Chataut   ||
====================================

Enter number of processes: 5
Enter process name, arrival time, and burst time:
A 0 10
B 5 15
C 2 3
D 6 4
E 5 9

Gantt Chart:
-------------------------------------------------
|   A   |   C   |   D   |   E   |   B   |
-------------------------------------------------
0      10      41      13      17      26

Process AT      BT      CT      TAT (CT-AT)     WT (TAT-BT)
-----------------------------------------------------------------
A       0       10      10      10 - 0 = 10     10 - 10 = 0
B       5       15      41      41 - 5 = 36     36 - 15 = 21
C       2       3       13      13 - 2 = 11     11 - 3 = 8
D       6       4       17      17 - 6 = 11     11 - 4 = 7
E       5       9       26      26 - 5 = 21     21 - 9 = 12

Average Turnaround Time = 17.80
Average Waiting Time = 9.60
```