Madan Bhandari Memorial College

Lab No. 6 **Date: 2025/09/01**

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

Introduction

Round Robin (RR) is a preemptive CPU scheduling algorithm where each process is assigned a fixed time slice or time quantum. The CPU executes each process in the ready queue for the duration of the quantum, then moves to the next process in a cyclic manner.

Characteristics:

- Preemptive: A process may be interrupted if it exceeds the time quantum.
- Fair: Every process gets an equal chance to execute.
- Suitable for time-sharing systems.

Algorithm Steps:

1. Initialize currentTime = 0.
2. Keep a ready queue of processes that have arrived.
3. Pick the first process in the queue and execute it for:
    - Time Quantum if remaining burst > quantum, or
    - Remaining burst time if ≤ quantum.
4. Update remaining burst time and currentTime.
5. If the process is not finished, move it to the end of the ready queue.
6. Repeat steps 3–5 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 remaining; // Remaining burst time
   int completion;
   int turnaround;
   int waiting;
};
int main()
{
   cout << "====================================" << endl;
   cout << "||    Round Robin Scheduling    ||" << endl;
   cout << "|| Compiled by :- Jonash Chataut  ||" << endl;
   cout << "====================================" << endl << endl;
   int n, quantum;
   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].remaining = p[i].burst;
      p[i].completion = 0;
   }
   cout << "Enter Time Quantum: ";
   cin >> quantum;
   int currentTime = 0, completed = 0;
   double totalTAT = 0, totalWT = 0;
   cout << "\nGantt Chart:\n";
   cout << "-------------------------------------------------\n";
   // RR Scheduling
   while (completed < n)
   {
      bool idle = true;
      for (int i = 0; i < n; i++)
      {
         if (p[i].remaining > 0 && p[i].arrival <= currentTime)
         {
            idle = false;
            cout << "| " << setw(4) << left << p[i].name << " ";

            if (p[i].remaining <= quantum)
```

```
        {
            currentTime += p[i].remaining;
            p[i].completion = currentTime;
            p[i].remaining = 0;
            completed++;
        }
        else
        {
            currentTime += quantum;
            p[i].remaining -= quantum;
        }
      }
    }
    if (idle)
    {
      cout << "| IDLE ";
      currentTime++;
    }
}
cout << "|\n";
cout << "-------------------------------------------------\n";
// Print Timeline
cout << "Timeline:\n0";
currentTime = 0;
int temp[n];
for (int i = 0; i < n; i++)
    temp[i] = p[i].burst;
completed = 0;
while (completed < n)
{
    bool idle = true;
    for (int i = 0; i < n; i++)
    {
      if (temp[i] > 0 && p[i].arrival <= currentTime)
      {
        idle = false;
        if (temp[i] <= quantum)
        {
            currentTime += temp[i];
            temp[i] = 0;
            completed++;
        }
        else
        {
            currentTime += quantum;
            temp[i] -= quantum;
        }
        cout << setw(5) << currentTime;
      }
    }
    if (idle)
```

```
        currentTime++;
    }
    // Print Process Table with calculations
    cout << "\n\nProcess\tAT\tBT\tCT\tTAT (CT-AT)\tWT (TAT-BT)\n";
    cout << "-----------------------------------------------------------------\n";
    for (int i = 0; i < n; i++)
    {
        p[i].turnaround = p[i].completion - p[i].arrival;
        p[i].waiting = p[i].turnaround - p[i].burst;
        totalTAT += p[i].turnaround;
        totalWT += p[i].waiting;
        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\third sem Jonash\cg\la
==================================
||     Round Robin Scheduling     ||
|| Compiled by :- Jonash Chataut  ||
==================================

Enter number of processes: 4
Enter process name, arrival time, and burst time:
A 0 5
B 3 10
C 2 4
D 7 6
Enter Time Quantum: 3

Gantt Chart:
-------------------------------------------------------------------------
| A    | B    | C    | D    | A    | B    | C    | D    | B    | B    |
-------------------------------------------------------------------------
Timeline:
03    6    9    12   14   17   18   21   24   25

Process AT      BT      CT      TAT (CT-AT)     WT (TAT-BT)
-----------------------------------------------------------------
A       0       5       14      14 - 0 = 14     14 - 5 = 9
B       3       10      25      25 - 3 = 22     22 - 10 = 12
C       2       4       18      18 - 2 = 16     16 - 4 = 12
D       7       6       21      21 - 7 = 14     14 - 6 = 8

Average Turnaround Time = 16.50
Average Waiting Time = 10.25
```