Bansilal Ramnath Agarwal Charitable Trust's

VISHWAKARMA INSTITUTE OF TECHNOLOGY, PUNE-37

(An Autonomous Institute of Savitribai Phule University)

Department of Artificial Intelligence & Data Science

Division	AI-A
Name	Mayank Dhananjay Kulkarni
Roll.no	78
PRN	12320056
Batch	B3

Title: Write a program to compute the finish time, turnaround time, and waiting time for the following algorithms:

a) First come First serve

b) Shortest Job First (Preemptive and Non Preemptive)

c) Priority (Preemptive and Non Preemptive)

d) Round robin

Code:-

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

vector<int> fcfs(vector<int> &cpu_burst) {
    int n = cpu_burst.size();
    vector<int> waiting_time(n, 0);
    int sum = 0;
    for(int i=1; i<n; i++){
        waiting_time[i] = sum + cpu_burst[i-1];
        sum += cpu_burst[i-1];
    }
    return waiting_time;
}

vector<int> sjf(vector<int> &cpu_burst) {
    int n = cpu_burst.size();
    priority_queue<pair<int, int>, vector<pair<int,int>>,
greater<pair<int,int>>> q;
    for(int i=0;i<n;i++){
        q.push({cpu_burst[i], i});
    }
    vector<int> waiting_time(n, 0);
    int sum = 0;
    while(!q.empty()){
        int process_number = q.top().second;
        int time = q.top().first;
        q.pop();
        waiting_time[process_number] = sum;
        sum += time;
    }
    return waiting_time;
}
```

```
vector<int> priority_scheduling(vector<int> &cpu_burst,
vector<int>&priority) {
   int n = cpu_burst.size();
   vector<int> waiting_time(n, 0);
   priority_queue< pair<int, pair<int,int>>, vector<pair<int, pair<int,int>>>,
greater<pair<int, pair<int,int>>>> q;
   for(int i=0; i<n; i++){
      q.push({priority[i], {cpu_burst[i], i}});
   }
   int sum = 0;
   while(!q.empty()){
      int time = q.top().second.first;
      int process_number = q.top().second.second;
      q.pop();
      waiting_time[process_number] = sum;
      sum += time;
   }
   return waiting_time;
}

vector<int> round_robin(vector<int> &cpu_burst, int quant) {
   int n = cpu_burst.size();
   queue<pair<int,int>> q;
   for(int i=0;i<n;i++){
      q.push({cpu_burst[i], i});
   }
   int sum = 0;
   vector<int> waiting_time(n, 0);
   vector<int> process_left(n, 0);
   while(!q.empty()){
      int process_number = q.front().second;
      int time = q.front().first;
      q.pop();
      waiting_time[process_number] = sum - process_left[process_number] +
waiting_time[process_number];
      if(time <= quant){
         sum += time;
         process_left[process_number] = sum;
      } else {
         time -= quant;
         sum += quant;
         process_left[process_number] = sum;
```

```
            q.push({time, process_number});
        }
    }
    return waiting_time;
}

void print_table(const vector<int>& cpu_burst, const vector<int>&
waiting_times, const vector<int>& turnaround_times) {
    int n = cpu_burst.size();
    cout << left << setw(10) << "Process" << setw(12) << "Burst Time" <<
setw(15) << "Waiting Time" << "Turnaround Time" << endl;
    cout << "-----------------------------------------------------------" << endl;
    for (int i = 0; i < n; i++) {
        cout << left << setw(10) << i+1 << setw(12) << cpu_burst[i] <<
setw(15) << waiting_times[i] << turnaround_times[i] << endl;
    }
    cout << endl;
}

int main() {
    int n;
    cout << "Enter number of Processes: ";
    cin >> n;

    vector<int> cpu_burst(n);
    for (int i = 0; i < n; i++) {
        cout << "Enter CPU burst of process " << i + 1 << ": ";
        cin >> cpu_burst[i];
    }

    // FCFS
    vector<int> waiting_times_fcfs = fcfs(cpu_burst);
    vector<int> turnaround_times_fcfs(n);
    for (int i = 0; i < n; i++) {
        turnaround_times_fcfs[i] = waiting_times_fcfs[i] + cpu_burst[i];
    }

    cout << "FCFS Scheduling:" << endl;
    print_table(cpu_burst, waiting_times_fcfs, turnaround_times_fcfs);

    // SJF
    vector<int> waiting_times_sjf = sjf(cpu_burst);
    vector<int> turnaround_times_sjf(n);
```

```
    for (int i = 0; i < n; i++) {
        turnaround_times_sjf[i] = waiting_times_sjf[i] + cpu_burst[i];
    }

    cout << "SJF Scheduling:" << endl;
    print_table(cpu_burst, waiting_times_sjf, turnaround_times_sjf);

    // Priority Scheduling
    vector<int> priority(n);
    for (int i = 0; i < n; i++) {
        cout << "Enter priority of process " << i + 1 << ": ";
        cin >> priority[i];
    }
    vector<int> waiting_times_priority = priority_scheduling(cpu_burst,
priority);
    vector<int> turnaround_times_priority(n);
    for (int i = 0; i < n; i++) {
        turnaround_times_priority[i] = waiting_times_priority[i] + cpu_burst[i];
    }

    cout << "Priority Scheduling:" << endl;
    print_table(cpu_burst, waiting_times_priority, turnaround_times_priority);

    // Round Robin
    int quant;
    cout << "Enter time quantum for Round Robin: ";
    cin >> quant;
    vector<int> waiting_times_rr = round_robin(cpu_burst, quant);
    vector<int> turnaround_times_rr(n);
    for (int i = 0; i < n; i++) {
        turnaround_times_rr[i] = waiting_times_rr[i] + cpu_burst[i];
    }

    cout << "Round Robin Scheduling:" << endl;
    print_table(cpu_burst, waiting_times_rr, turnaround_times_rr);

    return 0;
}
```

Output:

```
Enter number of Processes: 4
Enter CPU burst of process 1: 5
Enter CPU burst of process 2: 8
Enter CPU burst of process 3: 2
Enter CPU burst of process 4: 6
FCFS Scheduling:
Process   Burst Time  Waiting Time   Turnaround Time
-------------------------------------------------------------
1         5           0              5
2         8           5              13
3         2           13             15
4         6           15             21

SJF Scheduling:
Process   Burst Time  Waiting Time   Turnaround Time
-------------------------------------------------------------
1         5           2              7
2         8           13             21
3         2           0              2
4         6           7              13

Enter priority of process 1: 2
Enter priority of process 2: 1
Enter priority of process 3: 4
Enter priority of process 4: 3
Priority Scheduling:
Process   Burst Time  Waiting Time   Turnaround Time
-------------------------------------------------------------
1         5           8              13
2         8           0              8
3         2           19             21
4         6           13             19

Enter time quantum for Round Robin: 3
Round Robin Scheduling:
Process   Burst Time  Waiting Time   Turnaround Time
-------------------------------------------------------------
1         5           8              13
2         8           13             21
3         2           6              8
4         6           13             19
```