Name-Nissi Rathod

Class-TE-COM .B

Roll NO.-64

ASSIGNMENT NO.4

CODE :

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

struct Process {
  int id, at, bt, prio, ct, tat, wt, rt, remaining;
};

void printTable(vector<Process> p) {
  cout << "\nPID\tAT\tBT\tPR\tCT\tTAT\tWT";
  for (auto &x : p)
    cout << "\nP" << x.id << "\t" << x.at << "\t" << x.bt << "\t" << x.prio
      << "\t" << x.ct << "\t" << x.tat << "\t" << x.wt;
  cout << endl;
}

// ---------- FCFS ----------
void fcfs(vector<Process> p) {
  sort(p.begin(), p.end(), [](auto &a, auto &b){ return a.at < b.at; });
  int time = 0;
  for (auto &x : p) {
        time = max(time, x.at) + x.bt;

        x.ct = time;

        x.tat = x.ct - x.at;

        x.wt = x.tat - x.bt;

    }

    cout << "\n--- FCFS Scheduling ---";

    printTable(p);

}


// ---------- SJF (Preemptive) ----------

void sjf(vector<Process> p) {

    int n = p.size(), done = 0, time = 0;

    for (auto &x : p) x.remaining = x.bt;

    while (done < n) {

        int idx = -1, mn = 1e9;

        for (int i=0;i<n;i++)

            if (p[i].at <= time && p[i].remaining > 0 && p[i].remaining < mn)

                mn = p[i].remaining, idx = i;

        if (idx == -1) { time++; continue; }

        p[idx].remaining--; time++;

        if (p[idx].remaining == 0) {

            done++;

            p[idx].ct = time;

            p[idx].tat = p[idx].ct - p[idx].at;

            p[idx].wt = p[idx].tat - p[idx].bt;

        }
```

```
  }
  cout << "\n--- SJF (Preemptive) Scheduling ---";
  printTable(p);
}

// ---------- Priority (Non-Preemptive) ----------
void priorityNP(vector<Process> p) {
  int n = p.size(), time = 0, done = 0;
  vector<bool> vis(n,false);
  while (done < n) {
    int idx = -1, pr = 1e9;
    for (int i=0;i<n;i++)
      if (!vis[i] && p[i].at <= time && p[i].prio < pr)
        pr = p[i].prio, idx = i;
    if (idx == -1) { time++; continue; }
    time += p[idx].bt;
    p[idx].ct = time;
    p[idx].tat = p[idx].ct - p[idx].at;
    p[idx].wt = p[idx].tat - p[idx].bt;
    vis[idx] = true; done++;
  }
  cout << "\n--- Priority (Non-Preemptive) Scheduling ---";
  printTable(p);
}

// ---------- Round Robin (Preemptive) ----------
```

```
void roundRobin(vector<Process> p, int q) {
  int n = p.size(), time = 0;
  queue<int> rq;
  vector<bool> inq(n,false);
  for (auto &x : p) x.remaining = x.bt;
  sort(p.begin(), p.end(), [](auto &a, auto &b){ return a.at < b.at; });
  rq.push(0); inq[0] = true;
  while (!rq.empty()) {
    int i = rq.front(); rq.pop(); inq[i]=false;
    if (p[i].at > time) time = p[i].at;
    int exec = min(q, p[i].remaining);
    p[i].remaining -= exec; time += exec;
    for (int j=0;j<n;j++)
      if (!inq[j] && p[j].at <= time && p[j].remaining>0)
        rq.push(j), inq[j]=true;
    if (p[i].remaining>0) rq.push(i), inq[i]=true;
    else { p[i].ct = time; p[i].tat = p[i].ct - p[i].at; p[i].wt = p[i].tat - p[i].bt; }
  }
  cout << "\n--- Round Robin (q="<<q<<") Scheduling ---";
  printTable(p);
}

// ---------- MAIN ----------
int main() {
  int n;
  cout << "Enter number of processes: ";
```

```
    cin >> n;
    vector<Process> p(n);
    cout << "Enter AT BT Priority for each process:\n";
    for (int i=0;i<n;i++) {
        p[i].id=i+1;
        cin >> p[i].at >> p[i].bt >> p[i].prio;
    }
    fcfs(p);
    sjf(p);
    priorityNP(p);
    roundRobin(p,2);
}
```

OUTPUT:

```
--- FCFS Scheduling ---
PID   AT    BT    PR    CT    TAT   WT
P1    0     5     2     5     5     0
P2    1     3     1     8     7     4
P3    2     8     4     16    14    6
P4    3     6     3     22    19    13


--- SJF (Preemptive) Scheduling ---
PID   AT    BT    PR    CT    TAT   WT
P1    0     5     2     5     5     0
P2    1     3     1     8     7     4
```

```
P3    2     8     4     21    19    11
P4    3     6     3     27    24    18

--- Priority (Non-Preemptive) Scheduling ---
PID   AT    BT    PR    CT    TAT   WT
P2    1     3     1     4     3     0
P1    0     5     2     9     9     4
P4    3     6     3     15    12    6
P3    2     8     4     23    21    13

--- Round Robin (q=2) Scheduling ---
PID   AT    BT    PR    CT    TAT   WT
P1    0     5     2     17    17    12
P2    1     3     1     9     8     5
P3    2     8     4     25    23    15
P4    3     6     3     21    18    12
```