

Practical 8

Question 1. Write a CPP program to simulate the CPU scheduling algorithm MLQ(Multilevel Queue).

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  void bubbleSort(vector<int>&at, vector<int>&bt, vector<int>&pid, vector<int>&prio){
5      int n=at.size();
6      for(int i=0;i<n-1;i++){
7          for(int j=0;j<n-1;j++){
8              if(at[j]>at[j+1] || (at[j]==at[j+1] && pid[j]>pid[j+1])){
9                  swap(at[j],at[j+1]); swap(pid[j],pid[j+1]);
10                 swap(bt[j],bt[j+1]); swap(prio[j],prio[j+1]);
11             }
12         }
13     }
14 }
15
16 int getPID(vector<int>&bt, vector<int>&at, vector<int>&remProcess, int currTime, vector<int>&remTime,
17 priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>>pq;
18 for(int pid : remProcess){
19     int i = pid - 1;
20     // Process must be Priority 3 (Batch), Arrived, and Not Completed
21     if(at[i]<=currTime && remTime[i]>0 && prio[i] == 3){
22         pq.push({remTime[i], at[i]*1000 + pid});
23     }
24 }
25 if(pq.empty()) return -1;
26 return pq.top().second % 1000;
27 }
28
29 int main(){
30     cout<<"--- MLQ SCHEDULER ---"<<endl;
31     int n; cout<<"Enter number of processes: "; cin>>n;
32     vector<int>pid(n), at(n), bt(n), prio(n), ct(n), tat(n), wt(n);
33     vector<float>pr(n);
34     for(int i=0;i<n;i++){ cout<<"AT P"<<i+1<<": "; cin>>at[i]; pid[i]=i+1; }
35     for(int i=0;i<n;i++){ cout<<"BT P"<<i+1<<": "; cin>>bt[i]; }
36     for(int i=0;i<n;i++){ cout<<"Prio P"<<i+1<<" (1=Sys, 2=Int, 3=Bat): "; cin>>prio[i]; }
37
38     bubbleSort(at, bt, pid, prio);
39
40     vector<int>remTime(bt);
41     vector<int>remProcess; for(int x:pid) remProcess.push_back(x);
42     vector<int>slice(n,0);
43     vector<bool>in_q(n, false);
44
45     queue<int> q0, q1; // Store Indices directly
46     int completed=0, currTime=at[0];
47
48     // Initial load
49     for(int i=0; i<n; i++){
50         if(at[i]<=currTime){
```

```

51         if(prio[i]==1) { q0.push(i); in_q[i]=true; }
52         else if(prio[i]==2) { q1.push(i); in_q[i]=true; }
53     }
54 }
55
56 while(completed<n){
57     int idx = -1, type = 0; // 1=Q0, 2=Q1, 3=Q2
58
59     // Priority Check
60     if(!q0.empty()){ idx=q0.front(); type=1; }
61     else if(!q1.empty()){ idx=q1.front(); type=2; }
62     else {
63         int id = getPID(bt, at, remProcess, currTime, remTime, prio);
64         if(id!=-1) {
65             for(int i=0;i<n;i++) if(pid[i]==id) idx=i;
66             type=3;
67         }
68     }
69
70     if(idx != -1){
71         // Execution
72         remTime[idx]--;
73         currTime++;
74         if(type==1 || type==2) slice[idx]++;
75
76         // ADD NEW ARRIVALS (Critical for P2 before P1 logic)
77         for(int i=0; i<n; i++){
78             if(at[i] == currTime && remTime[i]>0 && prio[i]!=3 && !in_q[i]){
79                 if(prio[i]==1) { q0.push(i); in_q[i]=true; }
80                 else if(prio[i]==2) { q1.push(i); in_q[i]=true; }
81             }
82         }
83
84         if(remTime[idx] == 0){
85             completed++;
86             ct[idx] = currTime;
87             if(type==1) q0.pop();
88             else if(type==2) q1.pop();
89             // Q2 handled by getPID
90         }
91         else {
92             // Check Quantum or Preemption
93             bool drop = false;
94             if(type==1 && slice[idx]==4) drop=true;
95             if(type==2 && slice[idx]==8) drop=true;
96
97             // Preemption Logic: If I am Q1, and Q0 has something now
98             if(type==2 && !q0.empty()) drop=true;
99             // Note: If preempted, slice is NOT reset in strict RR, but simplified here we often
100             // If preempted, we should reset slice if we want "fresh" quantum next time, or keep
101             // Standard MLQ: Preemption puts you back at tail.
102
103             if(drop){
104                 if(type==1) { q0.pop(); q0.push(idx); slice[idx]=0; }
105                 else if(type==2) { q1.pop(); q1.push(idx); slice[idx]=0; }
106             }

```

```

107     }
108 } else {
109     currTime++;
110     for(int i=0; i<n; i++){
111         if(at[i] == currTime && remTime[i]>0 && prio[i]!=3 && !in_q[i]){
112             if(prio[i]==1) { q0.push(i); in_q[i]=true; }
113             else if(prio[i]==2) { q1.push(i); in_q[i]=true; }
114         }
115     }
116 }
117 }
118
119 // Output
120 cout<<"PID\tPrio\tAT\tBT\tCT\tTAT\tWT\tPR"<<endl;
121 int sumtat=0, sumwt=0; float sumpr=0;
122 for(int i=0;i<n;i++){
123     tat[i]=ct[i]-at[i]; wt[i]=tat[i]-bt[i]; pr[i]=(float)tat[i]/bt[i];
124     cout<<pid[i]<<"\t"<<prio[i]<<"\t"<<at[i]<<"\t"<<bt[i]<<"\t"<<ct[i]<<"\t"<<tat[i]<<"\t"<<wt[i]
125     sumtat+=tat[i]; sumwt+=wt[i]; sumpr+=pr[i];
126 }
127 cout<<"Avg TAT: "<<(float)sumtat/n<<endl;
128 cout<<"Avg WT: "<<(float)sumwt/n<<endl;
129 cout<<"Avg PR: "<<sumpr/n<<endl;
130 }

```

```

--- MLQ SCHEDULER ---
Enter number of processes: 5
AT P1: 0
AT P2: 1
AT P3: 2
AT P4: 3
AT P5: 4
BT P1: 5
BT P2: 4
BT P3: 3
BT P4: 2
BT P5: 1
Prio P1 (1=Sys, 2=Int, 3=Bat): 3
Prio P2 (1=Sys, 2=Int, 3=Bat): 1
Prio P3 (1=Sys, 2=Int, 3=Bat): 2
Prio P4 (1=Sys, 2=Int, 3=Bat): 1
Prio P5 (1=Sys, 2=Int, 3=Bat): 2

```

PID	Prio	AT	BT	CT	TAT	WT	PR
1	3	0	5	15	15	10	3
2	1	1	4	5	4	0	1
3	2	2	3	10	8	5	2.66667
4	1	3	2	7	4	2	2
5	2	4	1	11	7	6	7

```

Avg TAT: 7.6
Avg WT: 4.6
Avg PR: 3.13333

```

FIGURE 1. MLQ example

Question 2. Write a CPP program to simulate the CPU scheduling algorithm MLFQ(Multilevel Feedback Queue).

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  void bubbleSort(vector<int>&at, vector<int>&bt, vector<int>&pid){
5      int n=at.size();
6      for(int i=0;i<n-1;i++){
7          for(int j=0;j<n-1;j++){
8              if(at[j]>at[j+1] || (at[j]==at[j+1] && pid[j]>pid[j+1])){
9                  swap(at[j],at[j+1]); swap(pid[j],pid[j+1]); swap(bt[j],bt[j+1]);
10             }
11         }
12     }
13 }
14
15 // SRTF Logic checks if process is in Q2 (Level 2)
16 int getPID(vector<int>&bt, vector<int>&at, vector<int>&remProcess, int currTime, vector<int>&remTime,
17             priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>>pq;
18             for(int pid : remProcess){
19                 int i = pid - 1;

```

```

20         if(at[i]<=currTime && remTime[i]>0 && level[i] == 2){
21             pq.push({remTime[i], at[i]*1000 + pid});
22         }
23     }
24     if(pq.empty()) return -1;
25     return pq.top().second % 1000;
26 }
27
28 int main(){
29     cout<<"--- MLFQ SCHEDULER ---"<<endl;
30     int n; cout<<"Enter number of processes: "; cin>>n;
31     vector<int>pid(n), at(n), bt(n), ct(n), tat(n), wt(n);
32     vector<float>pr(n);
33     for(int i=0;i<n;i++){ cout<<"AT P"<<i+1<<": "; cin>>at[i]; pid[i]=i+1; }
34     for(int i=0;i<n;i++){ cout<<"BT P"<<i+1<<": "; cin>>bt[i]; }
35
36     bubbleSort(at, bt, pid);
37
38     vector<int>remTime(bt);
39     vector<int>remProcess; for(int x:pid) remProcess.push_back(x);
40     vector<int>slice(n,0);
41     vector<int>level(n,0); // 0=Q0, 1=Q1, 2=Q2
42     vector<bool>in_q(n, false);
43
44     queue<int> q0, q1;
45     int completed=0, currTime=at[0];
46
47     // Initial load (All go to Q0)
48     for(int i=0; i<n; i++){
49         if(at[i]<=currTime){ q0.push(i); in_q[i]=true; }
50     }
51
52     while(completed<n){
53         int idx = -1, type = 0;
54
55         if(!q0.empty()){ idx=q0.front(); type=0; }
56         else if(!q1.empty()){ idx=q1.front(); type=1; }
57         else {
58             int id = getPID(bt, at, remProcess, currTime, remTime, level);
59             if(id!=-1) { for(int i=0;i<n;i++) if(pid[i]==id) idx=i; type=2; }
60         }
61
62         if(idx != -1){
63             remTime[idx]--;
64             currTime++;
65             if(type==0 || type==1) slice[idx]++;
66
67             // ADD NEW ARRIVALS to Q0
68             for(int i=0; i<n; i++){
69                 if(at[i] == currTime && remTime[i]>0 && !in_q[i]){
70                     q0.push(i); in_q[i]=true;
71                 }
72             }
73
74             if(remTime[idx] == 0){
75                 completed++;

```

```

76         ct[idx] = currTime;
77         if(type==0) q0.pop();
78         else if(type==1) q1.pop();
79     }
80     else {
81         // Logic: Pop, Demote, Push
82         bool demote = false;
83         if(type==0 && slice[idx]==4) {
84             q0.pop();
85             level[idx]=1; slice[idx]=0;
86             q1.push(idx); // Move to Q1
87         }
88         else if(type==1 && slice[idx]==8) {
89             q1.pop();
90             level[idx]=2; // Move to Q2 (SRTF)
91             // No push needed, getPID checks 'level' vector
92         }
93         else if(type==1 && !q0.empty()){
94             // Preemption: Q1 running, new Q0 process arrived
95             q1.pop();
96             q1.push(idx); // Put back in Q1
97             // Note: In strict MLFQ, we might keep slice or reset. Keeping usage is fair.
98         }
99     }
100 } else {
101     currTime++;
102     for(int i=0; i<n; i++){
103         if(at[i] == currTime && remTime[i]>0 && !in_q[i]){
104             q0.push(i); in_q[i]=true;
105         }
106     }
107 }
108 }
109
110 cout<<"PID\tAT\tBT\tCT\tTAT\tWT\tPR"<<endl;
111 int sumtat=0, sumwt=0; float sumpr=0;
112 for(int i=0;i<n;i++){
113     tat[i]=ct[i]-at[i]; wt[i]=tat[i]-bt[i]; pr[i]=(float)tat[i]/bt[i];
114     cout<<pid[i]<<"\t"<<at[i]<<"\t"<<bt[i]<<"\t"<<ct[i]<<"\t"<<tat[i]<<"\t"<<wt[i]<<"\t"<<pr[i]<<endl;
115     sumtat+=tat[i]; sumwt+=wt[i]; sumpr+=pr[i];
116 }
117 cout<<"Avg TAT: "<<(float)sumtat/n<<endl;
118 cout<<"Avg WT: "<<(float)sumwt/n<<endl;
119 cout<<"Avg PR: "<<sumpr/n<<endl;
120 }

```

```

--- MLFQ SCHEDULER ---
Enter number of processes: 5
AT P1: 0
AT P2: 2
AT P3: 3
AT P4: 4
AT P5: 5
BT P1: 15
BT P2: 13
BT P3: 19
BT P4: 21
BT P5: 5

```

PID	AT	BT	CT	TAT	WT	PR
1	0	15	57	57	42	3.8
2	2	13	54	52	39	4
3	3	19	64	61	42	3.21053
4	4	21	73	69	48	3.28571
5	5	5	53	48	43	9.6

```

Avg TAT: 57.4
Avg WT: 42.8
Avg PR: 4.77925

```

FIGURE 2. MLFQ example