

## Practical 6

**Question 1.** Write a CPP program to simulate the CPU scheduling algorithm Round Robin (RR)

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 void bubbleSort(vector<int>&at,vector<int>&bt,vector<int>&pid){
4     int n=at.size();
5     for(int i=0;i<n-1;i++){
6         for(int j=0;j<n-1;j++){
7             if(at[j]>at[j+1]){
8                 swap(at[j],at[j+1]);
9                 swap(pid[j],pid[j+1]);
10                swap(bt[j],bt[j+1]);
11            }
12        }
13    }
14 }
15 void readyQueue(queue<pair<int,int>>&rq,vector<bool>&remProcess,vector<int>&at,vector<int>&pid,int c
16     for(int i=0;i<at.size();i++){
17         if(at[i]<=currTime && !remProcess[i]){
18             rq.push({pid[i],i});
19             remProcess[i]=1;
20         }
21     }
22 }
23 int main(){
24     int n;
25     cout<<"Enter number of processes:"<<endl;
26     cin>>n;
27     vector<int>pid(n);
28     vector<int>at(n);
29     vector<int>bt(n);
30     vector<int>ct(n);
31     vector<int>tat(n);
32     vector<int>wt(n);
33     vector<float>pr(n);
34     vector<bool>remProcess(n,0);
35     queue<pair<int,int>>rq;
36     int tq;
37     cout<<"Enter Time quantum for round robin:";
38     cin>>tq;
39     cout<<endl;
40     for(int i=0;i<n;i++){
41         cout<<"Enter Arrival time of P"<<i+1<<endl;
42         int num;
43         cin>>num;
44         at[i]=num;
45         pid[i]=i+1;
46     }
47     for(int i=0;i<n;i++){
48         cout<<"Enter Burst time of P"<<i+1<<endl;
49         int num;
50         cin>>num;
```

```

51         bt[i]=num;
52     }
53     bubbleSort(at,bt,pid);
54     vector<int>remTime(bt);
55     int currTime=at[0];
56     int completed=0;
57     readyQueue(rq,remProcess,at,pid,currTime);
58     while(completed<n){
59         if(rq.empty()){
60             int nextArrival=INT_MAX;
61             for(int i=0;i<n;i++){
62                 if(!remProcess[i])nextArrival=min(nextArrival,at[i]);
63             }
64             if(nextArrival==INT_MAX)break;
65             currTime=nextArrival;
66             readyQueue(rq,remProcess,at,pid,currTime);
67         }
68         int getID=rq.front().first;
69         int index=rq.front().second;
70         rq.pop();
71         if(remTime[index]>tq){
72             remTime[index]-=tq;
73             currTime+=tq;
74             readyQueue(rq,remProcess,at,pid,currTime);
75             rq.push({getID,index});
76         }
77         else{
78             currTime+=remTime[index];
79             readyQueue(rq,remProcess,at,pid,currTime);
80             remTime[index]=0;
81             ct[index]=currTime;
82             completed++;
83         }
84     }
85 }
86 for(int i=0;i<n;i++){
87     tat[i]=ct[i]-at[i];
88     wt[i]=tat[i]-bt[i];
89     pr[i]=tat[i]*1.0/bt[i];
90 }
91 cout<<"PID\tAT\tBT\tCT\tTAT\tWT\tPR"<<endl;
92 for(int i=0;i<n;i++){
93     cout<<pid[i]<<"\t"<<at[i]<<"\t"<<bt[i]<<"\t"<<ct[i]<<"\t"<<tat[i]<<"\t"<<wt[i]<<"\t"<<pr[i]<<endl;
94 }
95 int sumtat=0,sumwt=0;
96 float sumpr=0;
97 for(int i=0;i<n;i++){
98     sumtat+=tat[i];
99     sumwt+=wt[i];
100    sumpr+=pr[i];
101 }
102 cout<<"Average Turn Around Time:"<<1.0*sumtat/n<<endl;
103 cout<<"Average Waiting Time:"<<1.0*sumwt/n<<endl;
104 cout<<"Average Penalty Ratio:"<<sumpr/n<<endl;
105 }
```

```

Enter number of processes:
3
Enter Time quantum for round robin:2

Enter Arrival time of P1
0
Enter Arrival time of P2
2
Enter Arrival time of P3
1
Enter Burst time of P1
3
Enter Burst time of P2
1
Enter Burst time of P3
2
PID      AT        BT        CT        TAT        WT        PR
1          0         3         6         6         3         2
3          1         2         4         3         1         1.5
2          2         1         5         3         2         3
Average Turn Around Time:4
Average Waiting Time:2
Average Penalty Ratio:2.16667

```

FIGURE 1. RR example output

**Question 2.** Write a CPP program to simulate the CPU scheduling algorithm Priority Scheduling.  
a) Non Preemptive

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int getPID(vector<int>&pr,vector<int>&at,vector<int>&remProcess,int currTime){
4     int n=remProcess.size();
5     priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>>pq;
6     for(int i=0;i<n;i++){
7         int pid=remProcess[i];
8         if(at[pid-1]<=currTime)pq.push({pr[pid-1],at[pid-1]*1000+pid});
9     }
10    if(pq.empty())return -1;
11    int encoded=pq.top().second;
12    return encoded%1000;
13 }
14
15 int main(){
16     int n;
17     cout<<"Enter number of processes:"<<endl;
18     cin>>n;
19     vector<int>pid(n);
20     vector<int>at(n);

```

```

21     vector<int>bt(n);
22     vector<int>pr(n);
23     vector<int>ct(n);
24     vector<int>tat(n);
25     vector<int>wt(n);
26     vector<float>penalty(n);
27     vector<int>remProcess(n);
28     for(int i=0;i<n;i++){
29         cout<<"Enter Arrival time of P"<<i+1<<endl;
30         int num;
31         cin>>num;
32         at[i]=num;
33         pid[i]=i+1;
34         remProcess[i]=i+1;
35     }
36     for(int i=0;i<n;i++){
37         cout<<"Enter Burst time of P"<<i+1<<endl;
38         int num;
39         cin>>num;
40         bt[i]=num;
41     }
42     for(int i=0;i<n;i++){
43         cout<<"Enter Priority of P"<<i+1<<endl;
44         int num;
45         cin>>num;
46         pr[i]=num;
47     }
48     int currTime=*min_element(at.begin(),at.end());
49     while(!remProcess.empty()){
50         int getID=getPID(pr,at,remProcess,currTime);
51         if(getID== -1){
52             currTime++;
53             continue;
54         }
55         for(int i=0;i<remProcess.size();i++){
56             if(remProcess[i]==getID){
57                 remProcess.erase(remProcess.begin()+i);
58                 break;
59             }
60         }
61         ct[getID-1]=currTime+bt[getID-1];
62         currTime+=bt[getID-1];
63     }
64     for(int i=0;i<n;i++){
65         tat[i]=ct[i]-at[i];
66         wt[i]=tat[i]-bt[i];
67         penalty[i]=tat[i]*1.0/bt[i];
68     }
69     cout<<"PID\tAT\tBT\tCT\tTAT\tWT\tPR\tPriority"<<endl;
70     for(int i=0;i<n;i++){
71         cout<<pid[i]<<"\t"<<at[i]<<"\t"<<bt[i]<<"\t"<<ct[i]<<"\t"<<tat[i]<<"\t"<<wt[i]<<"\t"<<penalt
72     }
73     int sumtat=0,sumwt=0;
74     float sumpenalty=0;
75     for(int i=0;i<n;i++){
76         sumtat+=tat[i];

```

```

77     sumwt+=wt[i];
78     sumpenalty+=penalty[i];
79 }
80 cout<<"Average Turn Around Time:"<<1.0*sumtat/n<<endl;
81 cout<<"Average Waiting Time:"<<1.0*sumwt/n<<endl;
82 cout<<"Average Penalty Ratio:"<<sumpenalty/n<<endl;
83 }
```

```

Enter number of processes:
3
Enter Arrival time of P1
0
Enter Arrival time of P2
1
Enter Arrival time of P3
2
Enter Burst time of P1
3
Enter Burst time of P2
1
Enter Burst time of P3
2
Enter Priority of P1
3
Enter Priority of P2
2
Enter Priority of P3
1
PID      AT       BT       CT       TAT       WT       PR       Priority
1         0        3        3        3        0        1        3
2         1        1        6        5        4        5        2
3         2        2        5        3        1        1.5      1
Average Turn Around Time:3.66667
Average Waiting Time:1.66667
Average Penalty Ratio:2.5
```

FIGURE 2. Priority Scheduling Non Preemptive example

### b)Preemptive

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 int getPID(vector<int>&pr,vector<int>&at,vector<int>&remProcess,int currTime){
4     int n=remProcess.size();
5     priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>>pq;
6     for(int i=0;i<n;i++){
7         int pid=remProcess[i];
8         if(at[pid-1]<=currTime)pq.push({pr[pid-1],at[pid-1]*1000+pid});
9     }
10    if(pq.empty())return -1;
11    int encoded=pq.top().second;
```

```

12     return encoded%1000;
13 }
14
15 int main(){
16     int n;
17     cout<<"Enter number of processes:"<<endl;
18     cin>>n;
19     vector<int>pid(n);
20     vector<int>at(n);
21     vector<int>bt(n);
22     vector<int>pr(n);
23     vector<int>ct(n);
24     vector<int>tat(n);
25     vector<int>wt(n);
26     vector<float>penalty(n);
27     vector<int>remProcess(n);
28     vector<int>remTime(n);
29     for(int i=0;i<n;i++){
30         cout<<"Enter Arrival time of P"<<i+1<<endl;
31         int num;
32         cin>>num;
33         at[i]=num;
34         pid[i]=i+1;
35         remProcess[i]=i+1;
36     }
37     for(int i=0;i<n;i++){
38         cout<<"Enter Burst time of P"<<i+1<<endl;
39         int num;
40         cin>>num;
41         bt[i]=num;
42         remTime[i]=num;
43     }
44     for(int i=0;i<n;i++){
45         cout<<"Enter Priority of P"<<i+1<<endl;
46         int num;
47         cin>>num;
48         pr[i]=num;
49     }
50     int currTime=*min_element(at.begin(),at.end());
51     while(!remProcess.empty()){
52         int getID=getPID(pr,at,remProcess,currTime);
53         if(getID== -1){
54             currTime++;
55             continue;
56         }
57         remTime[getID-1]--;
58         if(remTime[getID-1]==0){
59             for(int i=0;i<remProcess.size();i++){
60                 if(remProcess[i]==getID){
61                     remProcess.erase(remProcess.begin()+i);
62                     break;
63                 }
64             }
65             ct[getID-1]=currTime+1;
66         }
67         currTime++;

```

```

68 }
69 for(int i=0;i<n;i++){
70     tat[i]=ct[i]-at[i];
71     wt[i]=tat[i]-bt[i];
72     penalty[i]=tat[i]*1.0/bt[i];
73 }
74 cout<<"PID\tAT\tBT\tCT\tTAT\tWT\tPR\tPriority"=>>endl;
75 for(int i=0;i<n;i++){
76     cout<<pid[i]<<"\t"<<at[i]<<"\t"<<bt[i]<<"\t"<<ct[i]<<"\t"<<tat[i]<<"\t"<<wt[i]<<"\t"<<penalty[i]<<endl;
77 }
78 int sumtat=0,sumwt=0;
79 float sumpenalty=0;
80 for(int i=0;i<n;i++){
81     sumtat+=tat[i];
82     sumwt+=wt[i];
83     sumpenalty+=penalty[i];
84 }
85 cout<<"Average Turn Around Time:"<<1.0*sumtat/n<<endl;
86 cout<<"Average Waiting Time:"<<1.0*sumwt/n<<endl;
87 cout<<"Average Penalty Ratio:"<<sumpenalty/n<<endl;
88 }

```

```

Enter number of processes:
3
Enter Arrival time of P1
0
Enter Arrival time of P2
2
Enter Arrival time of P3
1
Enter Burst time of P1
3
Enter Burst time of P2
1
Enter Burst time of P3
2
Enter Priority of P1
1
Enter Priority of P2
3
Enter Priority of P3
2
PID      AT       BT       CT       TAT       WT       PR       Priority
1        0        3        3        3        0        1        1
2        2        1        6        4        3        4        3
3        1        2        5        4        2        2        2
Average Turn Around Time:3.66667
Average Waiting Time:1.66667
Average Penalty Ratio:2.33333

```

FIGURE 3. Priority Scheduling Preemptive example