Lab No	:	09
Name of the Lab	:	Implementation of Priority Scheduling Algorithm
ID	:	IT-17005

Objectives:

i) What is Priority Scheduling Algorithm?
ii) How to implementation in C?

Answer no (i):

Priority scheduling :

In case of priority scheduling the priority is not always set as the inverse of the CPU burst time, rather it can be internally or externally set, but yes the scheduling is done on the basis of priority of the process where the process which is most urgent is processed first, followed by the ones with lesser priority in order.

Processes with same priority are executed in FCFS manner.

The priority of process, when internally defined, can be decided based on memory requirements, time limits, number of open files, ratio of I/O burst to CPU burst etc.

Priority scheduling can be of two types:

1. ***Preemptive Priority Scheduling***: If the new process arrived at the ready queue has a higher priority than the currently running process, the CPU is preempted, which means the processing of the current process is stoped and the incoming new process with higher priority gets the CPU for its execution.
2. ***Non-Preemptive Priority Scheduling:*** In case of non-preemptive priority scheduling algorithm if a new process arrives with a higher priority than the current running process, the incoming process is put at the head of the ready queue, which means after the execution of the current process it will be processed.

1. ***Non Pre-emptive Priority Scheduling:***

Consider the below processes available in the ready queue for execution, with **arrival time** as 0 for all and given **burst times**.

Process	Burst time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

In Non-Preemptive priority scheduling highest priority process will be executed first. Hence the GANTT chart will be following:

P2	P5	P1	P3	P4

0 1 6 16 18 19

Process	A.T	B.T	W.T=(s.t-a.t) + (s.t-l.c.t)	T.A.T=B.T+W.T	C.T
P1	0	10	6	16	16
P2	0	1	0	1	1
P3	0	2	16	18	18
P4	0	1	18	19	19
P5	0	5	1	6	6

Average waiting time = $\frac{6+0+16+18+1}{5}$ = 8.2 ms

Average turn around time = $\frac{16+1+18+19+6}{5}$ = 12 ms

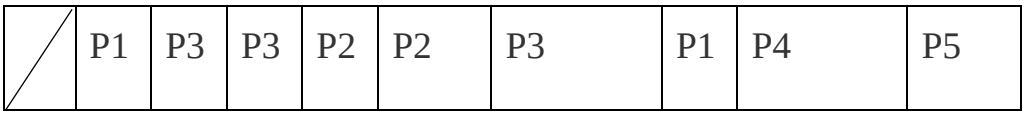
2. ***Pre-emptive Priority Scheduling:***

Consider the below processes available in the ready queue for execution, with **arrival time** as 0 for all and given **burst times**.

Process	Arrival time	Burst time	Priority
P1	1	2	4
P2	4	4	0
P3	2	10	2
P4	3	20	6
P5	5	2	8

In Preemptive priority scheduling highest priority process will be executed first.

Hence the GANTT chart will be following:



Process	A.T	B.T	W.T=(s.t-a.t) + (s.t-l.c.t)	T.A.T=B.T+W.T	R.T
P1	1	2	14	16	0
P2	4	4	0	4	0
P3	2	10	4	14	0
P4	3	20	14	34	14
P5	5	2	32	34	32

Average waiting time = $\frac{14+0+4+14+32}{5}$ = 12.8 ms

Average turn around time = $\frac{14+4+14+34+34}{5}$ = 20.4 ms

Answer no (ii):

The implementation of Preemptive Priority scheduling algorithm in C is given below:

Code:

```
//Implementation of preemptive priority scheduling
#include <iostream>
#include <algorithm>
#include <string.h>
using namespace std;

typedef struct proccess
{
   int at,bt,ct,ta,wt,btt,pr;
   string pro_id;
} schedule;

bool compare(schedule a,schedule b)
{
   return a.at<b.at;
}

bool compare2(schedule a,schedule b)
{
   return a.pr>b.pr;
}

int main()
{
   schedule pro[10];
   int n,i,j,pcom;
   double avg_wt,avg_tat,avg_ct,sum_wt=0,sum_tat=0,sum_ct=0;
   cout<<"Enter the number of process::";
   cin>>n;
```

```
for(i=0; i<n; i++)
{
cout<<"Enter the Process id, arrival time, burst time and priority :::";
   cin>>pro[i].pro_id>>pro[i].at>>pro[i].bt;
   pro[i].btt=pro[i].bt;
   cin>>pro[i].pr;
}

sort(pro,pro+n,compare);

pcom=0;
while(pcom<n)
{
   for(j=0; j<n; j++)
   {
      if(pro[j].at>i)
         break;
   }

   sort(pro,pro+j,compare2);

   if(j>0)
   {
      for(j=0; j<n; j++)
      {
         if(pro[j].bt!=0)
            break;
      }
      if(pro[j].at>i)
         i+=pro[j].at-i;
      pro[j].ct=i+1;
      pro[j].bt--;
   }

   i++;
   pcom=0;
   for(j=0; j<n; j++)
```

```
        {
            if(pro[j].bt==0)
                pcom++;
        }
    }

    cout<<"Process\tA.T\tB.T\tW.T\tT.A.T\tC.T\n";

    for(i=0; i<n; i++)
    {
        pro[i].ta=pro[i].ct-pro[i].at;
        pro[i].wt=pro[i].ta-pro[i].btt;

        sum_wt+=pro[i].wt;
        sum_tat+=pro[i].ta;
        sum_ct+=pro[i].ct;


cout<<pro[i].pro_id<<"\t"<<pro[i].at<<"\t"<<pro[i].btt<<"\t"<<pro[i]
.wt<<"\t"<<pro[i].ta<<"\t"<<pro[i].ct;
        cout<<endl;
    }
    avg_wt=sum_wt/n;
    avg_tat=sum_tat/n;
    avg_ct=sum_ct/n;

    cout<<"Average waiting time:"<<avg_wt<<endl;
    cout<<"Average turn around time:"<<avg_tat<<endl;
    cout<<"Average completion time:"<<avg_ct<<endl;
    return 0;
}
```

Output:

```
Enter the number of process::5
Enter the Process id, arrival time, burst time and priority :::p1 0 4 2
Enter the Process id, arrival time, burst time and priority :::p2 1 2 4
Enter the Process id, arrival time, burst time and priority :::p3 2 3 3
Enter the Process id, arrival time, burst time and priority :::p4 3 5 10
Enter the Process id, arrival time, burst time and priority :::p5 4 6 12
Process A.T     B.T     W.T     T.A.T   C.T
p5      4       6       1       7       11
p4      3       5       8       13      16
p2      1       2       15      17      18
p3      2       3       16      19      21
p1      0       4       21      25      25
Average waiting time:12.2
Average turn around time:16.2
Average completion time:18.2

Process returned 0 (0x0)   execution time : 55.579 s
Press any key to continue.
```