Lab Report No. **09**

Lab Report Name: **Implementation of Priority Scheduling algorithm .**

Objectives:

i What is Priority Scheduling algorithm.
ii How to implementation in C

Theory : In priority scheduling algorithm each process has a priority associated with it and as each process hits the queue, it is stored in based on its priority so that process with higher priority are dealt with first. It should be noted that equal priority processes are scheduled in FCFS order.

To prevent high priority processes from running indefinitely the scheduler may decrease the priority of the currently running process at each clock tick (i.e., at each clock interrupt). If this action causes its priority to drop below that of the next highest process, a process switch occurs. Alternatively, each process may be assigned a maximum time quantum that it is allowed to run. When this quantum is used up, the next highest priority process is given a chance to run.

Turnaround Time = Completion Time - Arrival Time
Waiting Time = Turn Around Time - Burst Time

Characteristics of Priority Scheduling

1. A CPU algorithm that schedules processes based on priority.
2. It used in Operating systems for performing batch processes.
3. If two jobs having the same priority are READY, it works on a FIRST COME, FIRST SERVED basis.
4. In priority scheduling, a number is assigned to each process that indicates its priority level.
5. Lower the number, higher is the priority.
6. In this type of scheduling algorithm, if a newer process arrives, that is having a higher priority than the currently running process, then the currently running process is preempted.

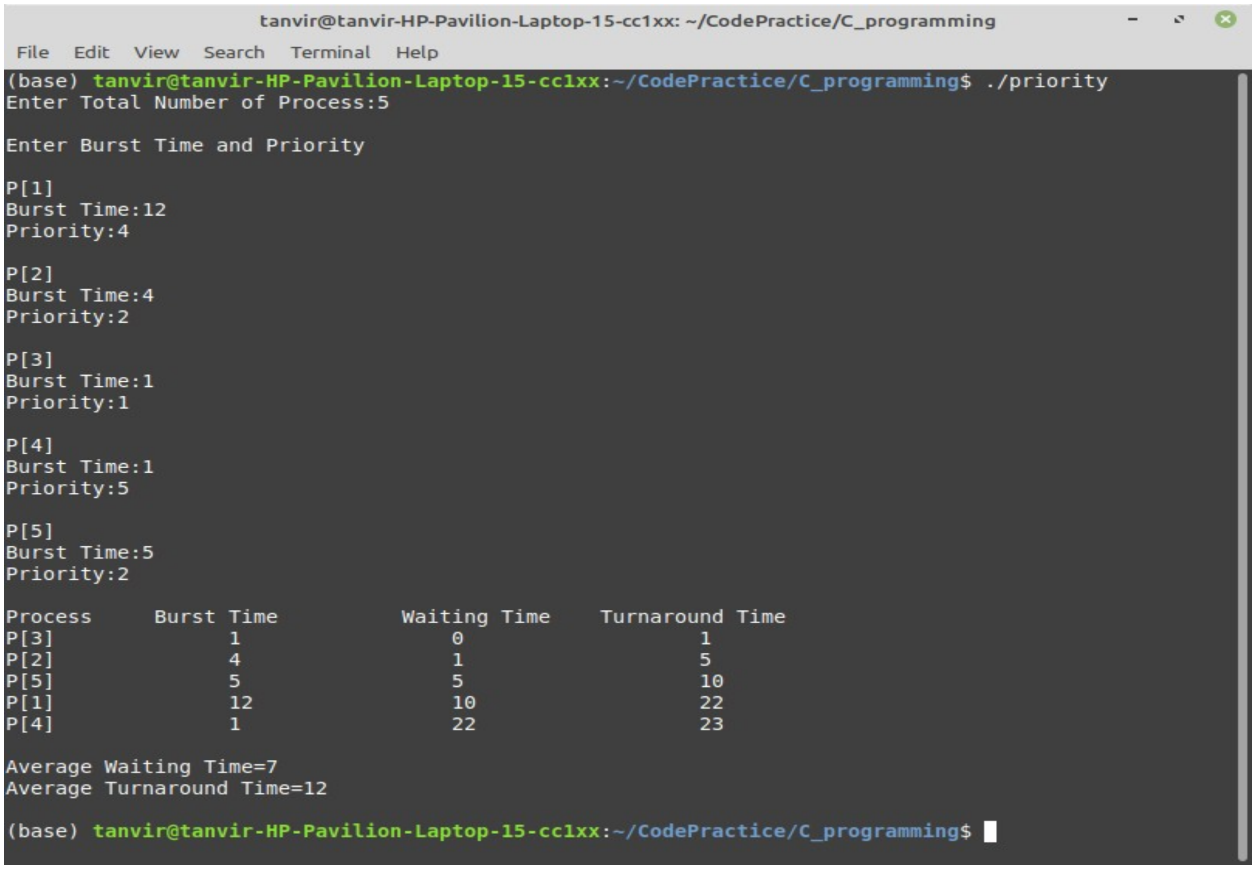
Corresponding Code:

```
#include<stdio.h>

int main()

{

   int bt[20], p[20], wt[20], tat[20], pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;

   printf("Enter Total Number of Process:");

   scanf("%d",&n);


   printf("\nEnter Burst Time and Priority\n");

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

   {

      printf("\nP[%d]\n",i+1);

      printf("Burst Time:");

      scanf("%d",&bt[i]);

      printf("Priority:");

      scanf("%d",&pr[i]);

      p[i]=i+1;          //contains process number

   }


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

   {

      pos=i;

      for(j=i+1; j<n; j++)

      {

         if(pr[j]<pr[pos])

            pos=j;

      }


      temp=pr[i];
```

```
    pr[i]=pr[pos];
    pr[pos]=temp;

    temp=bt[i];
    bt[i]=bt[pos];
    bt[pos]=temp;

    temp=p[i];
    p[i]=p[pos];
    p[pos]=temp;
}

wt[0]=0;    //waiting time for first process is zero

//calculate waiting time
for(i=1; i<n; i++)
{
    wt[i]=0;
    for(j=0; j<i; j++)
        wt[i]+=bt[j];

    total+=wt[i];
}

avg_wt=total/n;      //average waiting time
total=0;

printf("\nProcess\t    Burst Time    \tWaiting Time\tTurnaround Time");
for(i=0; i<n; i++)
```

```
   {
      tat[i]=bt[i]+wt[i];     //calculate turnaround time
      total+=tat[i];
      printf("\nP[%d]\t\t  %d\t\t    %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
   }
   avg_tat=total/n;     //average turnaround time
   printf("\n\nAverage Waiting Time=%d",avg_wt);
   printf("\nAverage Turnaround Time=%d\n",avg_tat);
   printf("\n");
   return 0;
}
```

Output:


tanvir@tanvir-HP-Pavilion-Laptop-15-cc1xx: ~/CodePractice/C_programming
File Edit View Search Terminal Help

```
(base) tanvir@tanvir-HP-Pavilion-Laptop-15-cc1xx:~/CodePractice/C_programming$ ./priority
Enter Total Number of Process:5

Enter Burst Time and Priority

P[1]
Burst Time:12
Priority:4

P[2]
Burst Time:4
Priority:2

P[3]
Burst Time:1
Priority:1

P[4]
Burst Time:1
Priority:5

P[5]
Burst Time:5
Priority:2

Process      Burst Time          Waiting Time     Turnaround Time
P[3]               1                  0                   1
P[2]               4                  1                   5
P[5]               5                  5                   10
P[1]               12                 10                  22
P[4]               1                  22                  23

Average Waiting Time=7
Average Turnaround Time=12

(base) tanvir@tanvir-HP-Pavilion-Laptop-15-cc1xx:~/CodePractice/C_programming$ 
```