**LOVELY PROFESSIONAL UNIVERSITY**

**Academic Task-3 (Operating System)**

School of Computer Science and Engineering Faculty of Technology And Sciences

Course Code: CSE 316 Course Title: Operating System

# Student Name: Aman Deep Student ID: 11805784

**Email Address: [hack.uean7@gmail.com](mailto:atriwalarpit@gmail.com)**

**GitHub Link: [https://github.com/hackuean7/OS\_Assignment](https://github.com/hackuean7)**

**Code:**

**#include<stdio.h> #include<conio.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;**

**}**

**//using sorting 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 of 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);**

**Problem:**

**return 0; getch();**

**}**

Design a scheduler that uses a preemptive priority scheduling algorithm based on dynamically changing priority. Larger number for priority indicates higher priority.

Assume that the following processes with arrival time and service time wants to execute (for reference):

|  |  |  |
| --- | --- | --- |
| **Process ID** | **Arrival Time** | **Service Time** |
| **P1** | **0** | **4** |
| **P2** | **1** | **1** |
| **P3** | **2** | **2** |
| **P4** | **3** | **1** |

When the process starts execution (i.e. CPU assigned), priority for that process changes at the rate of m=1.When the process waits for CPU in the ready queue (but not yet started execution), its priority changes at a rate n=2. All the processes are initially assigned priority value of 0 when they enter ready queue for the first time . The time slice for each process is q

= 1. When two processes want to join ready queue simultaneously, the process which has not executed recently is given priority. Calculate the average waiting time for each process. The program must be generic i.e. number of processes, their burst time and arrival time must be entered by user.

**Answer:**

According to questions statement, priority will be assigned on the basis of occurrence of process in queue ,so that priority increases by 1 in processes.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PID | Arrival Time | Burst Time | Completion Time | Turn Around Time | Waiting Time | Priority(Initially) | Priority(Under Process) |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | 0 | 4 | 8 | 8 | 4 | 0 | 1 |
| P2 | 1 | 1 | 2 | 1 | 1 | 0 | 2 |
| P3 | 2 | 2 | 5 | 3 | 3 | 0 | 3 |
| P4 | 3 | 1 | 4 | 1 | 3 | 0 | 4 |

Queue

We can also represent this example using GANTT chart as below:-

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P3 | P1 |

0 1 2 3 4 5 8

**Algorithm:**

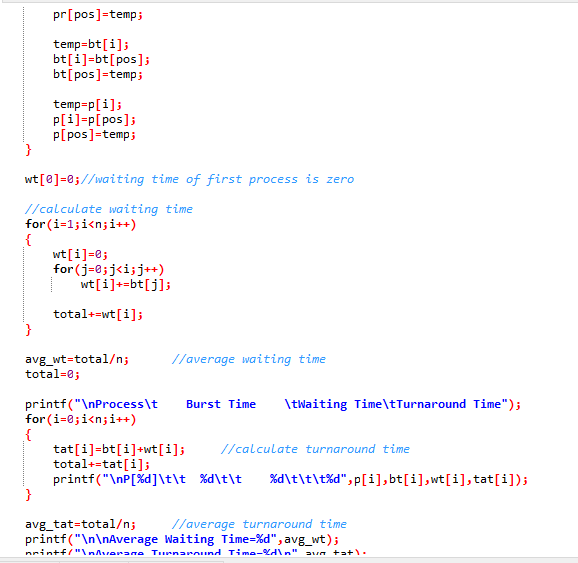
* In priority scheduling algorithm each process has a priority associated with it and as each process hits the queue, it is stored in based on its priority so that process with higher priority are dealt with first. It should be noted that equal priority processes are scheduled in [FCFS](https://www.thecrazyprogrammer.com/2014/11/c-cpp-program-for-first-come-first-served-fcfs.html) 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.

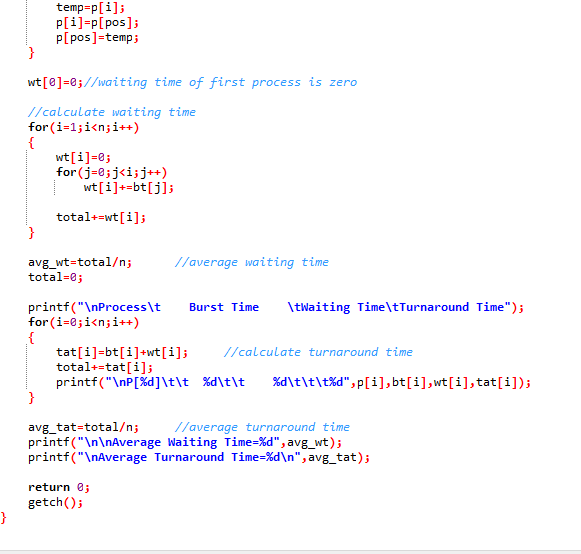
# Complexity:

**Overall Complexity of Above Program: O(n2 +n + n+ n)  O(n2 )**

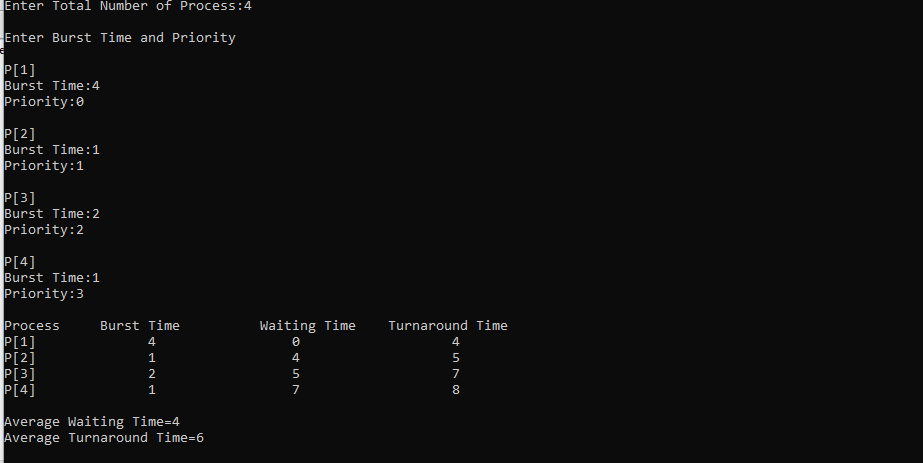
**Code Snippet:**



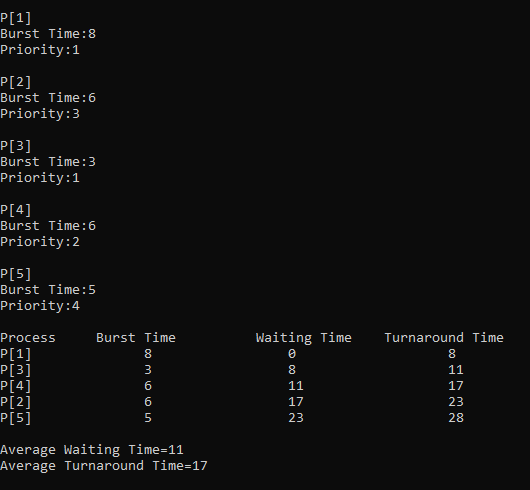




**Test Case 1:**



**Test Case 2:**



**Test Case 3:**

