Lovely Professional University

CSE-316

Operating Systems

Assignment simulation based

Name of the Student: R Ganesh

Regd no : 11711657

Roll no : 54

Section : EE034

Email id : ganesh181099@gmail.com

Ques no : 12

GitHub Link : https://github.com/11711657/Ganesh?files=1

Question: Ten students (a,b,c,d,e,f,g,h,i,j) are going to get there pictured clicked by university camera. Only one student can enter the camera room while the other students wait outside the room. The students are waiting in a queue to enter the room. To pass time the students start to play a game. In this game the students give candies to each other in a random manner (assume the students never run out of candies). They decide that the student with highest candies will be allowed to enter. When the student with highest amount of candies enter the room, the student starts the game again. Initially the students do not know if there is any body in the room and they start their game and the student with highest candies enter. Write and implement the algorithm to schedule such and compute the waiting and turnaround time. Consider the arrival time and burst time as given by the user.

Solution:

```
int main()

{

        char p[10][5],temp[5];

        int i,j,pt[10],wt[10],totwt=0,totat=0,pr[10],pa[10],ex[10],tat[10],temp1,n,sum=0;

        printf("Enter no of students:");
```

```
scanf("%d",&n);
for(i=0;i<n;i++)
{
        printf("enter student %d name:",i+1);
        scanf("%s",&p[i]);
        printf("enter arrival time:");
        scanf("%d",&pa[i]);
        printf("enter burst time:");
        scanf("%d",&pt[i]);
        printf("enter no of candies:");
        scanf("%d",&pr[i]);

}
for(i=0;i<n-1;i++)
{
        for(j=i+1;j<n;j++)
        {
                if(pr[i]>pr[j])
                {
                        temp1=pr[i];
                        pr[i]=pr[j];
                        pr[j]=temp1;
                        temp1=pt[i];
                        pt[i]=pt[j];
                        pt[j]=temp1;
                        strcpy(temp,p[i]);

                        strcpy(p[i],p[j]);

                        strcpy(p[j],temp);

                    }

                }

            }

                printf("S_name\t B_time\t A_time\t No.Candies\t ex_time\t W_time\t
        tat_time\n");

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

                {

                    sum+=pt[i];

                    ex[i]+=sum;

                    tat[i]=ex[i]-pa[i];

                totat+=tat[i];

                wt[i]=tat[i]-pt[i];

                totwt=totwt+wt[i];

                printf(" %s\t %d\t %d\t \t%d\t %d\t %d\n" ,p[i],pt[i],pa[i],pr[i],ex[i],wt[i],tat[i]);

                }


            printf("S_name\t B_time\t A_time\t No.Candies\t ex_time\t W_time\t tat_time\n");

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

            {

              printf(" %s\t %d\t %d\t \t%d\t %d\t %d\n" ,p[i],pt[i],pa[i],pr[i],ex[i],wt[i],tat[i]);

            }

            printf("total waiting time=%d\nturn around time=%d",totwt,totat);
```

getch();

}

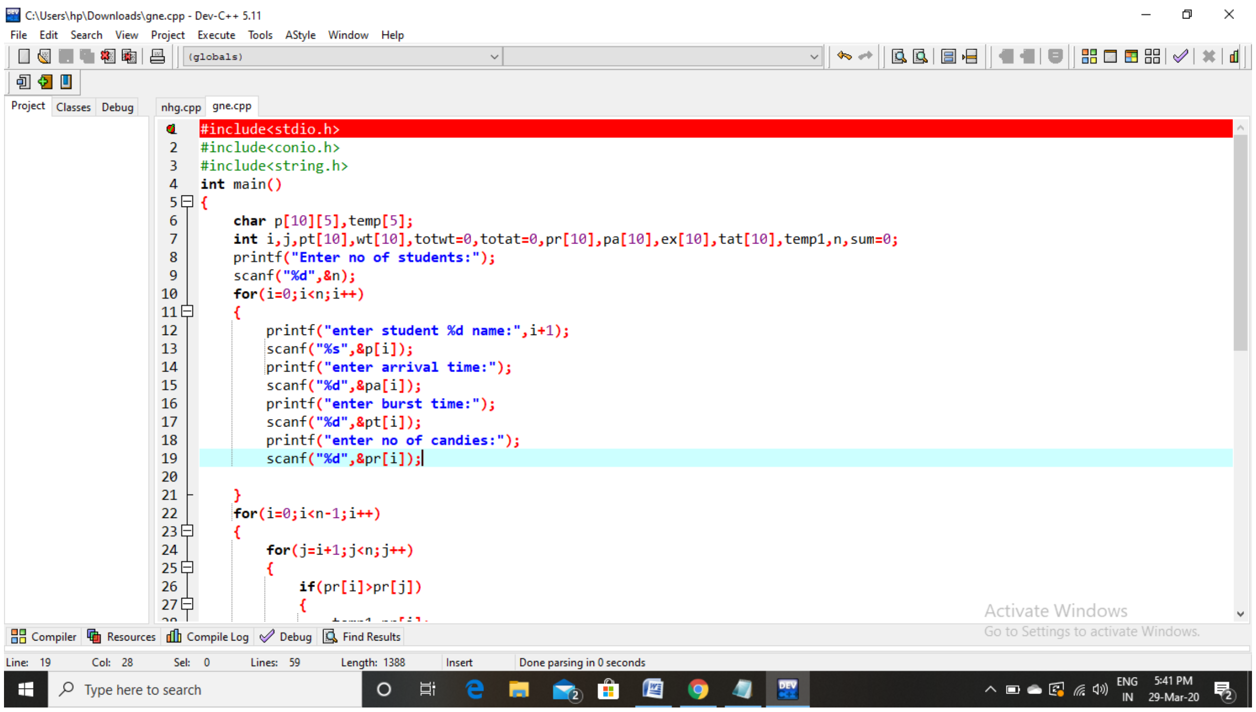
Code for the given question:

#include<stdio.h>

#include<conio.h>

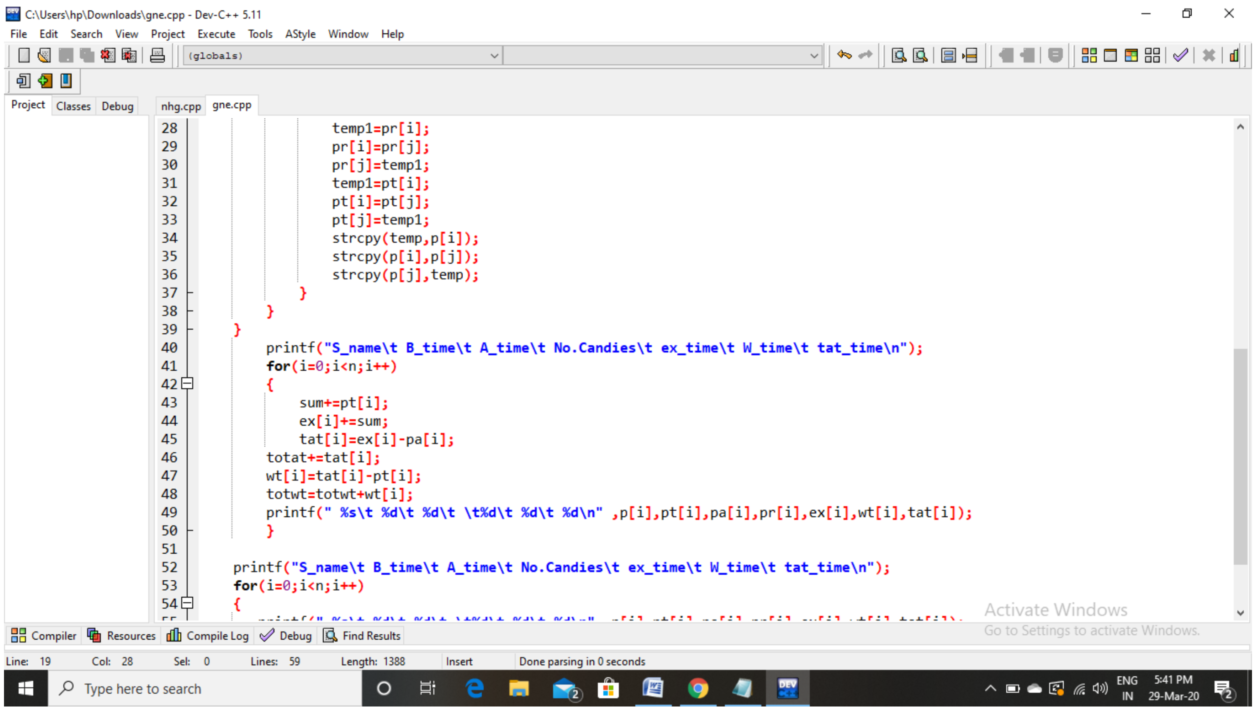
#include<string.h>

Screenshot:



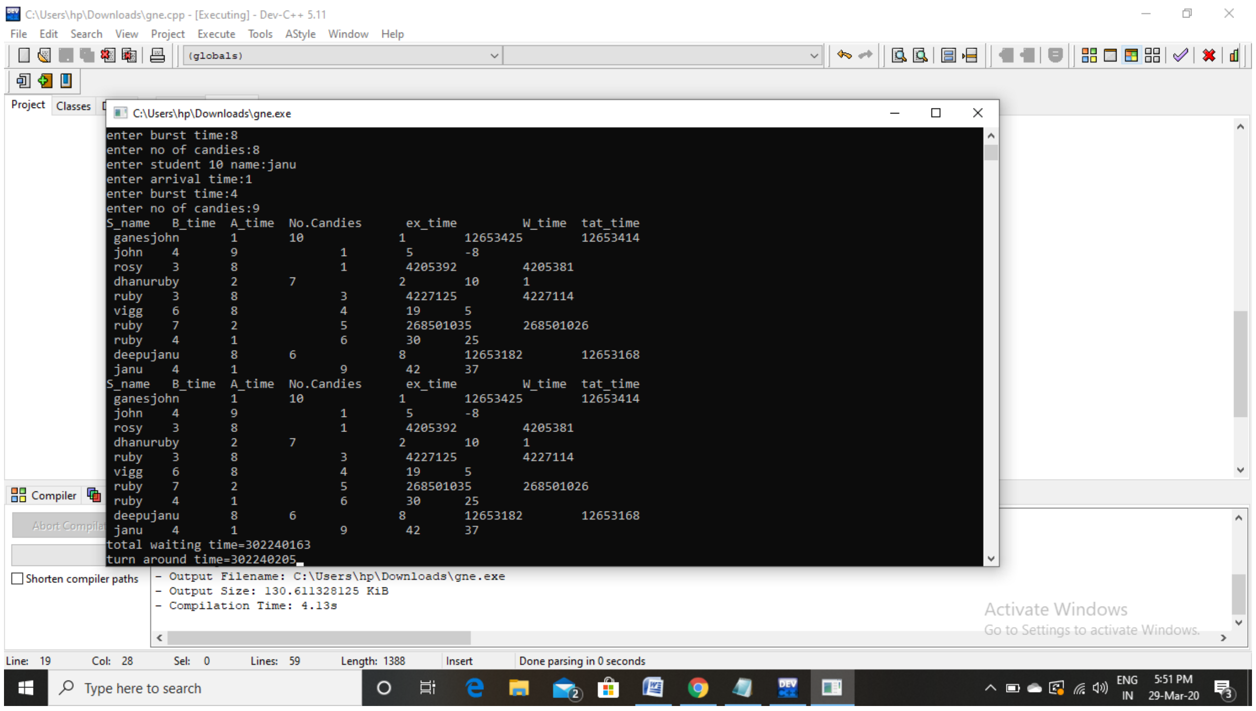
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int main()
{
    char p[10][5],temp[5];
    int i,j,pt[10],wt[10],totwt=0,totat=0,pr[10],pa[10],ex[10],tat[10],temp1,n,sum=0;
    printf("Enter no of students:");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("enter student %d name:",i+1);
        scanf("%s",&p[i]);
        printf("enter arrival time:");
        scanf("%d",&pa[i]);
        printf("enter burst time:");
        scanf("%d",&pt[i]);
        printf("enter no of candies:");
        scanf("%d",&pr[i]);

    }
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(pr[i]>pr[j])
            {
```



```
                temp1=pr[i];
                pr[i]=pr[j];
                pr[j]=temp1;
                temp1=pt[i];
                pt[i]=pt[j];
                pt[j]=temp1;
                strcpy(temp,p[i]);
                strcpy(p[i],p[j]);
                strcpy(p[j],temp);
            }
        }
    }
        printf("S_name\t B_time\t A_time\t No.Candies\t ex_time\t W_time\t tat_time\n");
        for(i=0;i<n;i++)
        {
            sum+=pt[i];
            ex[i]+=sum;
            tat[i]=ex[i]-pa[i];
        totat+=tat[i];
        wt[i]=tat[i]-pt[i];
        totwt=totwt+wt[i];
        printf(" %s\t %d\t %d\t \t%d\t %d\t %d\n" ,p[i],pt[i],pa[i],pr[i],ex[i],wt[i],tat[i]);
        }

    printf("S_name\t B_time\t A_time\t No.Candies\t ex_time\t W_time\t tat_time\n");
    for(i=0;i<n;i++)
    {
```



```
enter burst time:8
enter no of candies:8
enter student 10 name:janu
enter arrival time:1
enter burst time:4
enter no of candies:9
S_name   B_time  A_time  No.Candies      ex_time         W_time  tat_time
 ganesjohn       1       10              1       12653425        12653414
 john    4       9               1        5       -8
 rosy    3       8               1        4205392         4205381
 dhanuruby       2       7               2       10      1
 ruby    3       8               3        4227125         4227114
 vigg    6       8               4        19      5
 ruby    7       2               5        268501035       268501026
 ruby    4       1               6        30      25
 deepujanu       8       6               8       12653182        12653168
 janu    4       1               9        42      37
S_name   B_time  A_time  No.Candies      ex_time         W_time  tat_time
 ganesjohn       1       10              1       12653425        12653414
 john    4       9               1        5       -8
 rosy    3       8               1        4205392         4205381
 dhanuruby       2       7               2       10      1
 ruby    3       8               3        4227125         4227114
 vigg    6       8               4        19      5
 ruby    7       2               5        268501035       268501026
 ruby    4       1               6        30      25
 deepujanu       8       6               8       12653182        12653168
 janu    4       1               9        42      37
total waiting time=302240163
turn around time=302240205
```

```
- Output Filename: C:\Users\hp\Downloads\gne.exe
- Output Size: 130.611328125 KiB
- Compilation Time: 4.13s
```

Description:

This question relates with the Priority Based Scheduling algorithm. The process of assigning processes to the CPU based on their priority is known as Priority Based Scheduling algorithm. In this each process is associated with a priorty number(integer). The CPU is allocated to the process with highest priority. By default low number value means higher prority.

Ex: 0-> highest priority

1-> lowest priority

Problem:

The problem we are facing in this type of scheduling is “Starvation”.

It is a condition where a process does not get the required resources for a long time because the resources are being allocated to other processes. The lower priority process may never executes.

Ex: P1-> low priority

P2-> higher priority then here p1 has to wait if suppose p3 also has higher priority then p1 should keep on waiting for execution.

Solution:

The problem in this can be avoided by the process of “Aging”. It is a scheduling technique used to avoid starvation in priority scheduling. It is a process of gradually increasing the priority of a task by basing on its waiting time in the ready state. I previous example we have seen that if p2 and p3 has high priority than p1 then p1 undergoes aging in the ready state respectively. And then the priority of p1 will increase gradually. So the lower priority of p1 will also executes without any violation.

Complexity:

It depends on the amount of time taken to execute the input and to give the output.

Turn around time = completion time – arrival time

Waiting time = turn around time – burst time

Conclusion:

Thus the given question is solved using priority based scheduling and the techniques required to avoid starvation.