Student Name : Gottumukkala Kanakaraju

Student ID : 11806904

Email Address : gkr5413@gmail.com

GitHub Link : https://github.com/gkr5413/OS/tree/master/os

GitHub Profile : https://github.com/gkr5413

1.EXPLANATION:

Q1.

Github Link for Problem 1: https://github.com/gkr5413/OS/blob/master/os/OS%208.c

Explanantion for Q1 : processes are scheduled using both preemptive, round robin scheduling algorithm too. Each processes are assignese a numeric priority, with higher number of indicatingrelative priority. In addition to process listed below,then the system is idle task. Then the task has priority 0 and the scheduled. when the system has no any other available process to run in memory.10 units is the length of time Quantum. If a process have preempted by a higher-priority processes, the preempted process are placed in the end of the queue.then Thread Priority Burst ArrivalWAP in C code to.What is the CPU utilization rate Show theprosessing sheduling by using a Gantt chart.and what is the turn around for each process what iswaiting time foreach process

Q2.

Github Link ForProblem 2:https://github.com/gkr5413/OS/blob/master/os/OS%2019v1.cpp

Explanation For Q2 : In the given problem of 10 students are purchasing the gift and returning to gift counter. The accountant gives preference to student who has maximum number of gifts. If we talk about this problem in Operating System concept, In priority scheduling algorithm, a priority is associated with each process and CPU is allocated to the process with the higher priority. If two processes have same priority then they will be execute in first come first serve order.

2.SOLUTIONS:

CODE FOR PROBLEM 1:

```
#include<stdio.h>
#include<conio.h>
int main()
 {
   int A , n,  C[10], Q[10], B[10], D[10], V[10], Z , X, i, Y[10], T;

   printf(" number of process ? : ");
   scanf("%d",&n);
   printf("time quantum");
   scanf("%d",&T);
   printf("\n\t burst time : time priorities : Arrival time  \n");

   for(i=0;i<n;i++)
    {
      printf("\n Process %d ",i+1);
      scanf("%d %d %d",&B[i],&Q[i],&Y[i]);
            C[i]=i+1;
    }
int j;
  for(i=0;i<n-1;i++)
   {
     for(j=i+1;j<n;j++)
     {
      if(Q[i]<Q[j])
       {
```

```
    A=Q[i];
    Q[i]=Q[j];
    Q[j]=A;
    A=B[i];
    B[i]=B[j];
    B[j]=A;
    A=C[i];
    C[i]=C[j];
    C[j]=A;
     }
  }
}
D[0]=0;
Z=0;
V[0]=B[0];
X=V[0];
for(i=1;i<n;i++)
 {
            if( T<n || T>n ) {


   D[i]=V[i-1];
   Z+=D[i];
   V[i]=D[i]+B[i];
   X+=V[i];
             }
 }
 printf("\n");
 printf(" Then Gantt chart is \n");
 for(i=0;i<n;i++)
 {
            printf("P %d ",C[i]);
 }
```

```
printf("\n\nProcess \t Burst Time \t Wait Time \t Turn Around Time   Priority \tArrival time
\n");
for(i=0;i<n;i++){
printf("\n  %d",C[i]); printf("\t\t %d",B[i]); printf("\t\t %d",D[i]); printf("\t\t %d",V[i]);
printf("\t\t %d",Q[i]); printf("\t\t %d",Y[i]);
}
Z/=n;
X/=n;
printf("\n Average Waiting Time : %d ",Z);
printf("\n Average TurnAround Time : %d",X);
getch();
}
```

CODE FOR PROBLEM 2:

```
#include<stdio.h>
struct student{
        int number_of_gift;
        int arrival_time;
        int id;
}
s[10];
void sort_arival(){
        struct student a;
        int min,pos;
        for(int i=0;i<10;i++){
                min = s[i].arrival_time;
                pos = i;
                for(int j=i+1;j<10;j++){
                        if(min>s[j].arrival_time){
                                min = s[j].arrival_time;
```

```
                                        pos = j;
                                }
                        }

                        if(i!=pos){
                                a=s[i];
                                s[i]=s[pos];
                                s[pos]=a;
                        }
                }
        }
int main(){

        for(int i=0;i<10;i++)
        {
                s[i].id=i+1;
                printf("\nEnter the arrival time of Students%d: ",i+1);
                scanf("%d",&s[i].arrival_time);
                printf("\nEnter the number of gift items for Students%d: ",i+1);
                scanf("%d",&s[i].number_of_gift);
        }
        sort_arival();
        struct student m;
        int pos;
        for(int i=0;i<10;i++){
                m = s[i];
                pos = i;
                for(int j=i+1;j<10;j++){
                        if(m.number_of_gift<s[j].number_of_gift){
                                m = s[j];
                                pos = j;
                        }
```

```
                    }
                    s[pos]=s[i];
                    s[i]=m;
                    printf("\nBill turn for student s%d having %d
items",s[i].id,s[i].number_of_gift);
          }

          return 0;
}
```

3.ALGORITHMS:

Problem 1 : it Does’t have any algorithm in use

Problem 2
I have used **Bubble Sort** algorithm in problem 2 to sort the processes in descending order based on the arrival time of processes and updated there priority and sorted the order of process based on there current priority.

COMPLEXITY:

Problem 1: The complexity is **N^2**

Problem 2: The complexity of **bubble sort** is **O(n2)** in both worst and average cases, because the entire array needs to be iterated for every element.

4.EXPLAIN CONSTRAINS:

Problem 2:

In this problems, we are given constraint of 10 students. Whereas I applied a constraint of gifts a student can purchase, i.e. 1 <= gift <50 This implies that one has to buy at least 1 gift or no student can buy more than 49 gifts.

5. ADDITIONAL ALGORITHM SUPPORT:

Bubble Sort the fastest sort available under a very specific circumstance. It originally became well known primarily because it was one of the first algorithms (of any kind) that was rigorously analyzed, and the proof was found that it was optimal under its limited circumstance

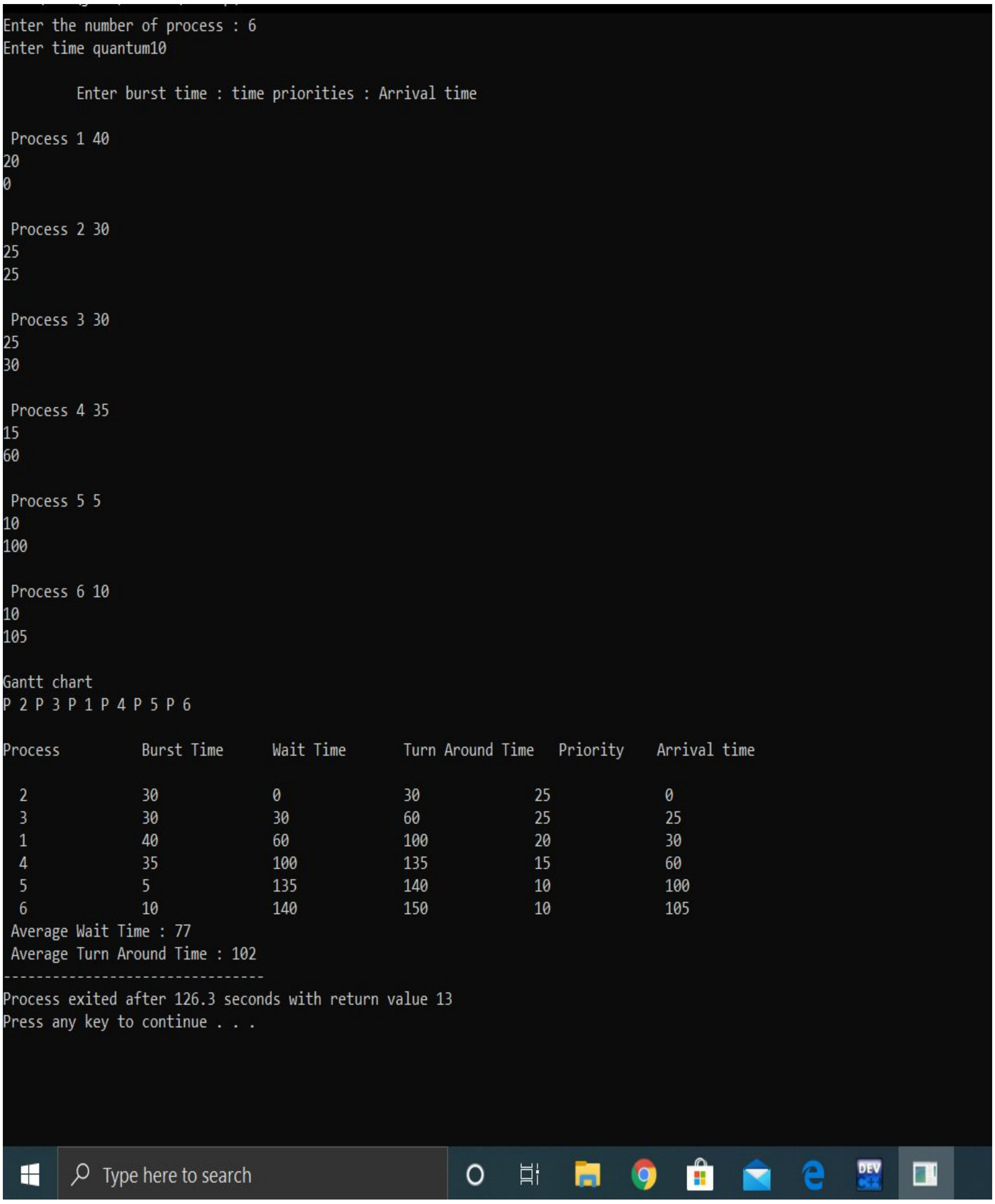
```
void sort_arival(){
        struct student a;
        int min,pos;
        for(int i=0;i<10;i++){
                min = s[i].arrival_time;
                pos = i;
                for(int j=i+1;j<10;j++){
                        if(min>s[j].arrival_time){
                        min = s[j].arrival_time;
                                pos = j;
                        }
                }
                if(i!=pos){
                        a=s[i];
                        s[i]=s[pos];
                        s[pos]=a;
                }
        }
}
```

6.BOUNDRY CONDITION:

Max value for the process should be small other wise long time to process will be take

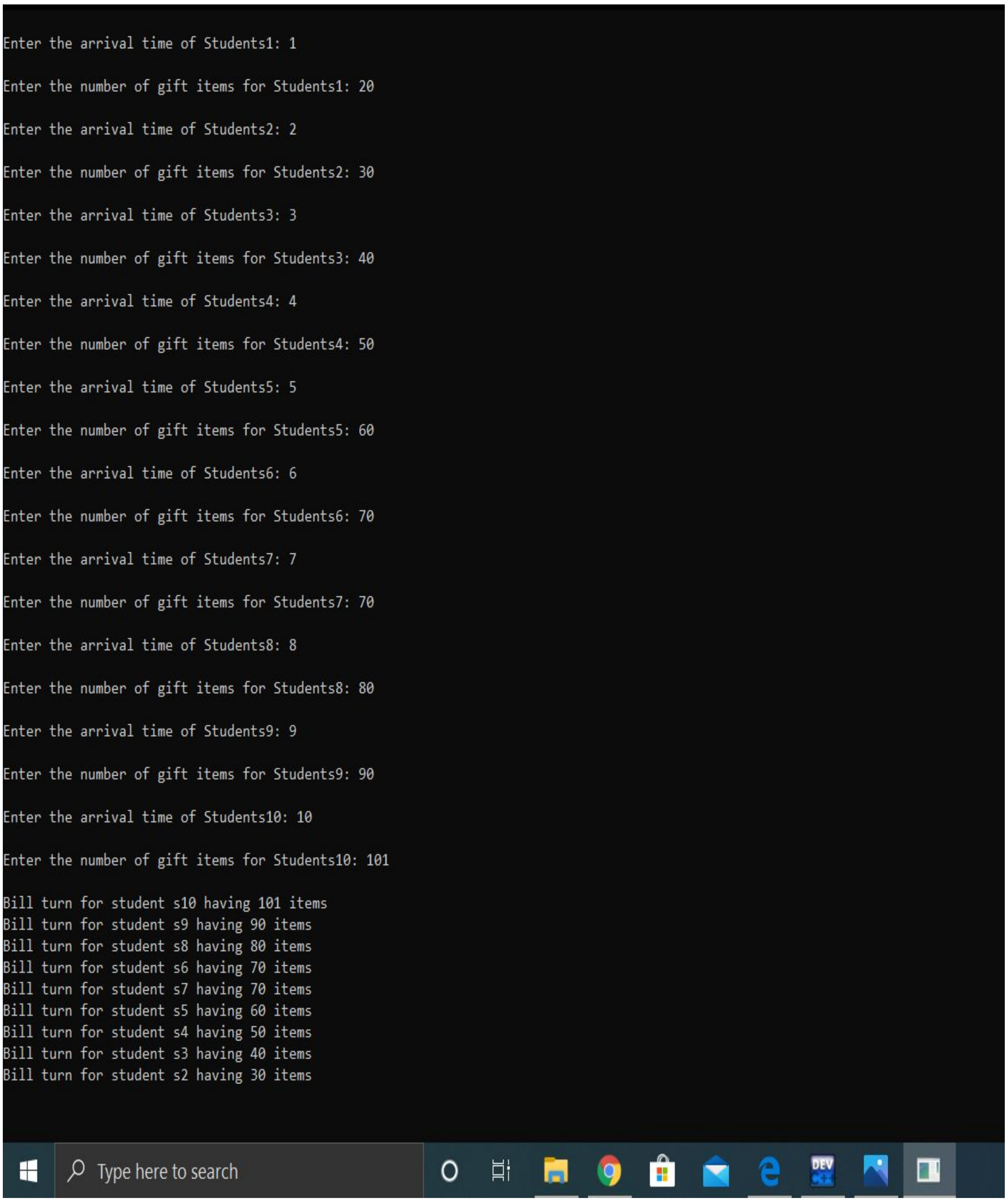
7.TEST CASE

OUTPUT FOR PROBLEM 1:



```
Enter the number of process : 6
Enter time quantum10

         Enter burst time : time priorities : Arrival time

 Process 1 40
20
0

 Process 2 30
25
25

 Process 3 30
25
30

 Process 4 35
15
60

 Process 5 5
10
100

 Process 6 10
10
105

Gantt chart
P 2 P 3 P 1 P 4 P 5 P 6

Process          Burst Time      Wait Time       Turn Around Time   Priority    Arrival time

  2              30              0               30              25              0
  3              30              30              60              25              25
  1              40              60              100             20              30
  4              35              100             135             15              60
  5              5               135             140             10              100
  6              10              140             150             10              105
 Average Wait Time : 77
 Average Turn Around Time : 102
--------------------------------
Process exited after 126.3 seconds with return value 13
Press any key to continue . . .
```

OUTPUT FOR PROBLEM 2:



```
Enter the arrival time of Students1: 1

Enter the number of gift items for Students1: 20

Enter the arrival time of Students2: 2

Enter the number of gift items for Students2: 30

Enter the arrival time of Students3: 3

Enter the number of gift items for Students3: 40

Enter the arrival time of Students4: 4

Enter the number of gift items for Students4: 50

Enter the arrival time of Students5: 5

Enter the number of gift items for Students5: 60

Enter the arrival time of Students6: 6

Enter the number of gift items for Students6: 70

Enter the arrival time of Students7: 7

Enter the number of gift items for Students7: 70

Enter the arrival time of Students8: 8

Enter the number of gift items for Students8: 80

Enter the arrival time of Students9: 9

Enter the number of gift items for Students9: 90

Enter the arrival time of Students10: 10

Enter the number of gift items for Students10: 101

Bill turn for student s10 having 101 items
Bill turn for student s9 having 90 items
Bill turn for student s8 having 80 items
Bill turn for student s6 having 70 items
Bill turn for student s7 having 70 items
Bill turn for student s5 having 60 items
Bill turn for student s4 having 50 items
Bill turn for student s3 having 40 items
Bill turn for student s2 having 30 items
```

8.GITHUB LINKS:

https://github.com/gkr5413/OS/tree/master/os For Whole Folder.

https://github.com/gkr5413/OS/blob/master/os/OS%208.c For Problem 1.

https://github.com/gkr5413/OS/blob/master/os/OS%2019v1.cpp For Problem 2.