Name: Aman Soni

Registration no: 11602961

Email id: ramero.am@gmail.com

GitHub link: https://github.com/Aman-1234/OS-Project

Code:

```
#include<stdio.h>
#include<string.h>
#include<pthread.h>
#include<stdlib.h>
#include<time.h>

int st_size=0,tc_size=0;
int chk=0;
int mini_avg_student=1000,stud_loc,mini_turn_reca=100;
int mini_avg_teacher=1000,teac_loc;
int quant=20,size;
int timer=0;

struct tm * time_information;

//structure for collection of all the details of the queue
struct que
{
   int priority_1;     //to differentiate b/w stent and teacher
   int burst_time;     //Burst_time
   int rem_btime;     //Remaining burst time
   int arri_time;    //arrival time of query
   char per_name[20];   //Name of the counsumer
   int turn_rec;       //repeation record
};
```

```
//st_que array for student queries
//tc_que array for teacher queries
struct que st_que[10],tc_que[10];

//function to get the system current time
void gettime()
{
   time_t rawtime;

   time ( &rawtime );
   time_information = localtime ( &rawtime );

}

//function to chk for the valid time to be loggeg in 1 for ture else 0
void chk_time()
{
   gettime();
   if(time_information->tm_hour>=10 && time_information->tm_hour<12)
   {
      chk=1;
   }
}


//function will print all the details of the queries
void print_data_1(struct que a[],int size)
{
   printf("sr_no.\tName\t\tPosition\tArrival_time\tRemaining_Burst_time\tturn_rec\n");
```

```
   char buff[20];
   if(a[0].priority_1==1)
      strcpy(buff,"Student");
   else
      strcpy(buff,"Teacher");

   for(int i=0;i<size;i++)
   {

      printf("%-6d  %-15s %-15s %-15d %-15d %-
10d\n",i+1,a[i].per_name,buff,a[i].arri_time,a[i].burst_time,a[i].turn_rec);
   }



}

//funtion will find the best nxt student query for execution check both arrival time , burst time and
repeation of querys
void pro_min_stent()
{mini_avg_student=1000;
   mini_turn_reca=10;
   for(int i=0;i<st_size;i++)
   {
      if(st_que[i].burst_time>0 && mini_turn_reca>st_que[i].turn_rec)
      if(mini_avg_student>=st_que[i].arri_time)
      {
            mini_avg_student=st_que[i].arri_time;
            stud_loc=i;
            mini_turn_reca=st_que[i].turn_rec;

           // printf("%d",stud_loc);
```

```
            //printf("yp\n");
        }


    }


   // cout<<st_que[stud_loc].per_name<<endl;
}


//funtion will find the best nxt teacher query for execution chk both arrival time , burst time and repeation of querys
void pro_mini_teach()
{
    mini_avg_teacher=1000;
    mini_turn_reca=10;
    for(int i=0;i<tc_size;i++)
    {
        if(tc_que[i].burst_time>0)
        if(mini_avg_teacher>=tc_que[i].arri_time && mini_turn_reca>st_que[i].turn_rec)
        {
            mini_avg_teacher=tc_que[i].arri_time;
            teac_loc=i;
            mini_turn_reca=tc_que[i].turn_rec;
            //printf("yoo\n");
        }


    }
    //cout<<tc_que[teac_loc].per_name<<endl;
}


//function will remove the specified node fromm the que array
```

```
void rem_elem(struct que * temp)
{
   int i=0;
   if(temp->priority_1==2)
   {
      while(&st_que[i]!=temp)
         i++;

      if(i!=st_size)
         while(i<st_size-1)
         {
            strcpy(st_que[i].per_name,st_que[i+1].per_name);
            st_que[i].arri_time=st_que[i+1].arri_time;
            st_que[i].burst_time=st_que[i+1].burst_time;
            i++;
         }
      //else
         st_size--;
   }
   if(temp->priority_1==1)
   {
      while(&tc_que[i]!=temp)
         i++;

      if(i!=tc_size)
         while(i<tc_size-1)
         {
            strcpy(tc_que[i].per_name,tc_que[i+1].per_name);
            tc_que[i].arri_time=tc_que[i+1].arri_time;
            tc_que[i].burst_time=tc_que[i+1].burst_time;
```

```
            i++;
          }
       //else
          tc_size--;
    }

}


//function will execute the que and update the que status
void *pro(struct que *temp)
{
    system("clear");
    printf("\" %s \" your turn is here \n",temp->per_name);

    sleep(4);
    printf("\n\nProcessing your request\n\n");
    sleep(4);
    if ((temp->burst_time > 0))
    {
       temp->burst_time -= quant;
       if (temp->burst_time <=0)
       {
          timer+=temp->burst_time+quant;
          temp->burst_time=0;
          printf("\" %s \" query is completly executed :\n",temp->per_name);
          rem_elem(temp);
          //size--;
       }
       else
```

```
        {
            timer+=quant;
            //printf("%d",temp->burst_time);
            printf("\nSorry for the inconvinance \n");
            printf("\" %s \" query is to big you Wait for Your next turn_rec\n",temp->per_name);
            temp->turn_rec++;

        }
        sleep(5);


    }
     pthread_exit(NULL);
}

//main function
int main()
{
    pthread_t p2;

    int Total_querytime=0,Query_count=0;
    float average_querytime=0;
    //chk_time();
    chk=1;
    system("clear");
    printf("Suresh Welcome To Online Query System:\n");
    sleep(3);
    system("clear");
    printf("Logging in .\n");
    sleep(2);
```

```
system("clear");
printf("Logging in . .\n");
sleep(2);
system("clear");
printf("Logging in . . .\n");


// chk the loggin in critaria
if(chk==1)
{
   int flag1=1;
   sleep(2);
   system("clear");
   printf("Logged In Succesfull\n");
   while(flag1)
   {
      char name[20],position[20];
      int arival_time,bursst_time;
      sleep(1);
      system("clear");
      //taking query from the user

      printf("\nEnter the  query details in the form:\n");
      printf("\nEnter your name: ");
      scanf("%s",name);
      printf("\nEnter your position(student/teacher): ");
      scanf("%s",position);
      printf("\nEnter your arrival time: ");
      scanf("%d",&arival_time);
      printf("\nEnter your query time needed: ");
```

```
scanf("%d",&bursst_time);

struct que *temp;

if(strcmp(position,"student")==0||strcmp(position,"STUDENT")==0)
{
   temp=&st_que[st_size];
   //printf("hello\n");
   st_size++;
   temp->priority_1=2;
}
else if(strcmp(position,"teacher")==0||strcmp(position,"TEACHER")==0)
{
   temp=&tc_que[tc_size];
   //printf("yoo\n");
   tc_size++;
   temp->priority_1=1;
}

temp->arri_time=arival_time;
temp->burst_time=bursst_time;
temp->rem_btime=bursst_time;
strcpy(temp->per_name,name);
temp->turn_rec=1;

Total_querytime+=bursst_time;
Query_count++;

printf("\n\nAdd another form(Y/N)\n");
```

```
   char ch;
   scanf("%c",&ch);
   scanf("%c",&ch);
   if(ch=='y'||ch=='Y')
   flag1=1;
   else
   flag1=0;


}
system("clear");

//printing of query table
printf("List Of the Student Query Submitted\n\n");
print_data_1(st_que,st_size);
printf("\n\nList Of the Teacher Query Submitted\n\n");
print_data_1(tc_que,tc_size);
//printf("%d\n",st_size);
//printf("%d\n",tc_size);

printf("NOTE:- Every Query will be given 20 time Quantum:-\n");

printf("\n\nWait till we call your name: \n\n ");

pro_min_stent();
pro_mini_teach();
sleep(6);

if(mini_avg_student<mini_avg_teacher )
   {
```

```
    timer=mini_avg_student;
  }
  else
  {
    timer=mini_avg_teacher;
  }

//calling query according to there priority_1 and arrival time
while(tc_size!=0||st_size!=0)
{

  //printf("%d %d \n",mini_avg_student,mini_avg_teacher);



  if(timer>=mini_avg_teacher)
  {
    pthread_create(&p2,NULL,pro,(void *)&tc_que[teac_loc]);
    pthread_join(p2,NULL);
    //st_size--;
  }
  else
  {
    pthread_create(&p2,NULL,pro,(void *)&st_que[stud_loc]);
    pthread_join(p2,NULL);
   // tc_size--;

  }

  //print_data_1(st_que,st_size);
```

```
            //print_data_1(tc_que,tc_size);
            //printf("%d\n",st_size);
            //printf("%d\n",tc_size);
            pro_min_stent();
            pro_mini_teach();

        }
        average_querytime=Total_querytime/Query_count;
        sleep(3);
        system("clear");
        printf("Todays query taking session has been Ended \n");
        printf("\n\nTotal query taken today = %d \n\nTotal query time = %d\n\nAverage query time =
%f\n",Query_count,Total_querytime,average_querytime);
        sleep(3);
        system("clear");
        printf("Logging out .\n");
        sleep(2);
        system("clear");
        printf("Logging out . .\n");
        sleep(2);
        system("clear");
        printf("Logging out . . .\n");
        sleep(2);
        system("clear");
        printf("Logged out\n");



    }
    else
```

```
    {
        system("clear");
        printf("log In Unsucessfull\n\n");
        printf("Cannt login during this time of day\n");

    }

}
```

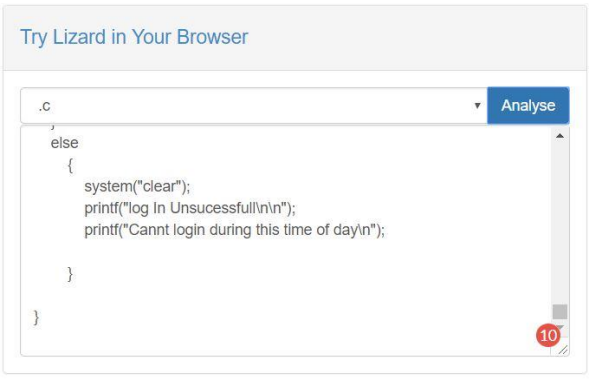
Problem explanation:

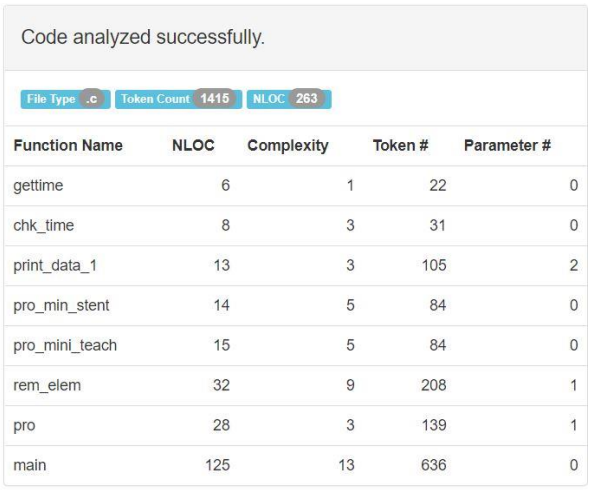
Sudesh Sharma is a Linux expert who wants to have an online system where he can handle student queries. Since there can be multiple requests at any time he wishes to dedicate a fixed amount of time to every request so that everyone gets a fair share of his time. He will log into the system from 10am to 12am only. He wants to have separate requests queues for students and faculty, where faculty quese is given a higher priority.

Also, the summary at the end of the session should include the total time he spent on handling queries and average query time.

In order to implement this scenario, following implementation approach has been taken.

Complexity:





Function Name	NLOC	Complexity	Token #	Parameter #
gettime	6	1	22	0
chk_time	8	3	31	0
print_data_1	13	3	105	2
pro_min_stent	14	5	84	0
pro_mini_teach	15	5	84	0
rem_elem	32	9	208	1
pro	28	3	139	1
main	125	13	636	0

Constraints:

1.Sudesh sharma can log into the system from 10am to 12am only.

```
//function to chk for the valid time to be loggeg in 1 for ture else 0
void chk_time()
{
    gettime();
    if(time_information->tm_hour>=10 && time_information->tm_hour<12)
    {
        chk=1;
    }
}
```

2.Sudesh want to have separate requests queues for students and faculty,

```
//structure for collection of all the details of the queue
struct que
{
    int priority_1;     //to differentiate b/w stent and teacher
    int burst_time;     //Burst_time
    int rem_btime;     //Remaining burst time
    int arri_time;    //arrival time of query
    char per_name[20];   //Name of the counsumer
    int turn_rec;       //repeation record
};

//st_que array for student queries
//tc_que array for teacher queries
struct que st_que[10],tc_que[10];
```

3. Teachers query is given a higher priority(lower the no. Heigher the priority)

```
if(strcmp(position,"student")==0||strcmp(position,"STUDENT")==0)
{
    temp=&st_que[st_size];
    //printf("hello\n");
    st_size++;
    temp->priority_1=2;
}
else if(strcmp(position,"teacher")==0||strcmp(position,"TEACHER")==0)
{
    temp=&tc_que[tc_size];
    //printf("yoo\n");
    tc_size++;
    temp->priority_1=1;
}
```

Additional Algorithm(Round-Robin algo):

Since there can be multiple requests at any time . So avoid starvation sudesh wishes to dedicate a fixed amount of time to every request. And to implement that i used Round-Robin algorithum to remove starvation with time quantom of 20unit .so that everyone gets a fair share of his time.

```
//function will execute the que and update the que status
void *pro(struct que *temp)
{
    system("clear");
    printf("\" %s \" your turn is here \n",temp->per_name);

    sleep(4);
    printf("\n\nProcessing your request\n\n");
    sleep(4);
    if ((temp->burst_time > 0))
    {
        temp->burst_time -= quant;
        if (temp->burst_time <=0)
        {
            timer+=temp->burst_time+quant;
            temp->burst_time=0;
            printf("\" %s \" query is completly executed :\n",temp->per_name);
            rem_elem(temp);
            //size--;
        }
        else
        {
            timer+=quant;
            //printf("%d",temp->burst_time);
            printf("\nSorry for the inconvinance \n");
            printf("\" %s \" query is to big you Wait for Your next turn_rec\n",temp->per_name);
            temp->turn_rec++;

        }
        sleep(5);


    }
     pthread_exit(NULL);
}
```

Boundary Condition:

1. Query having minimum arrival time will get the first chance (like FCFS) if they have the same position(Student/Teacher) .

2. Teacher Query will be giving more priority then Students if and only if the arrival time is equal or less than student.

3. If a query is not able to complete in the given time quantum then it is placed at the back of the que to remove starvation conditions and will get his chance of execution when all the remaining query are processed.

4. Student query will be handled if no teacher query has arrived i.e arrival time of teacher is greater than the time counter.

5. Only one query can be handled at a time for only 20 unit time quantum.

Test Case:

*__When Sudesh login between 10am to 12am;__

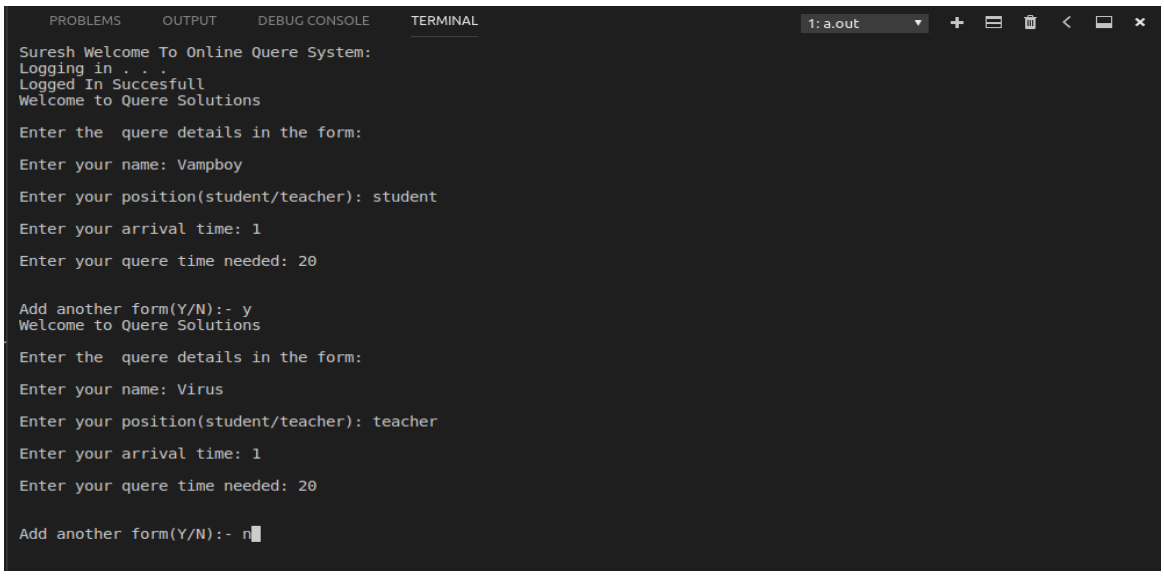
note: all the “clear” statements are commented to show all the output in one go because of that all the output screens are so close to each other

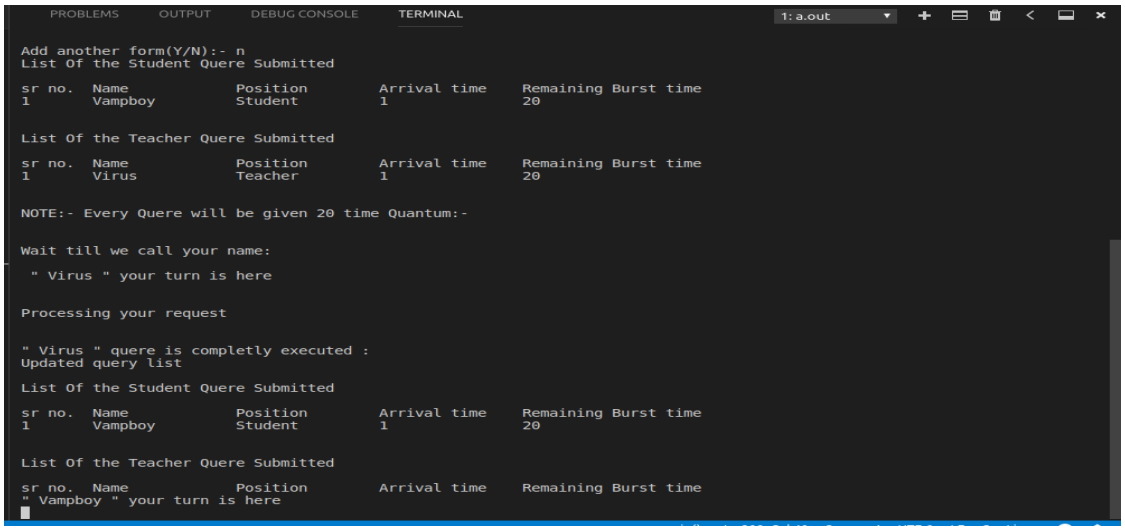
1. when both arrival time is same i.e. 1 and both query burst time is equal to time quantum 20 unit.

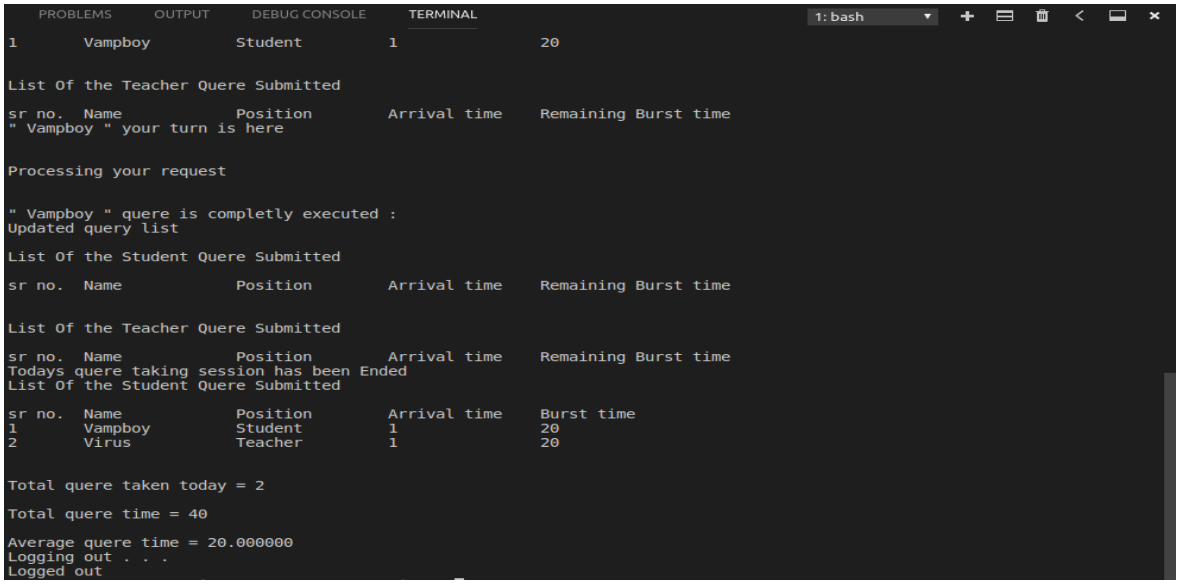
T1-completed	S1-completed

1 20 40

Description:- as both the S1 and T1 process arrive at the same time as the teacher process has the higher priority therefore it executes first and terminates after 20 time quantum and after that S1 will get the chance of Execution.


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: a.out
Suresh Welcome To Online Quere System:
Logging in . . .
Logged In Succesfull
Welcome to Quere Solutions
Enter the quere details in the form:
Enter your name: Vampboy
Enter your position(student/teacher): student
Enter your arrival time: 1
Enter your quere time needed: 20
Add another form(Y/N):- y
Welcome to Quere Solutions
Enter the quere details in the form:
Enter your name: Virus
Enter your position(student/teacher): teacher
Enter your arrival time: 1
Enter your quere time needed: 20
Add another form(Y/N):- n

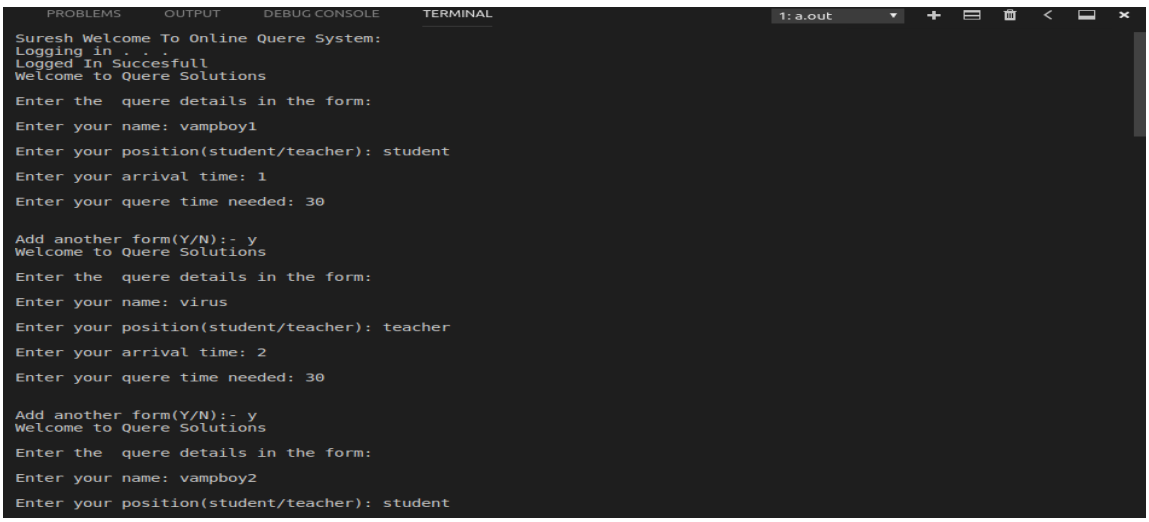

PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: a.out
Add another form(Y/N):- n
List Of the Student Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
1 Vampboy Student 1 20
List Of the Teacher Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
1 Virus Teacher 1 20
NOTE:- Every Quere will be given 20 time Quantum:-
Wait till we call your name:
" Virus " your turn is here
Processing your request
" Virus " quere is completly executed :
Updated query list
List Of the Student Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
1 Vampboy Student 1 20
List Of the Teacher Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
" Vampboy " your turn is here



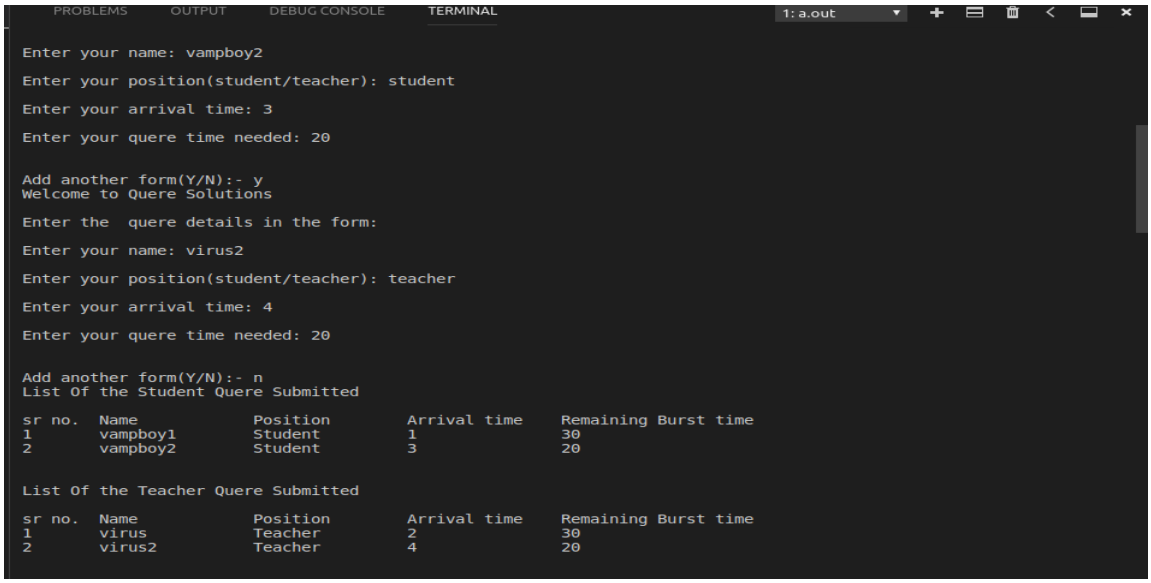
2. when arrival time of student1<teacher1<student2<teacher2 and with query burst time of 30,30,20,20 respectively. Execution be like :-

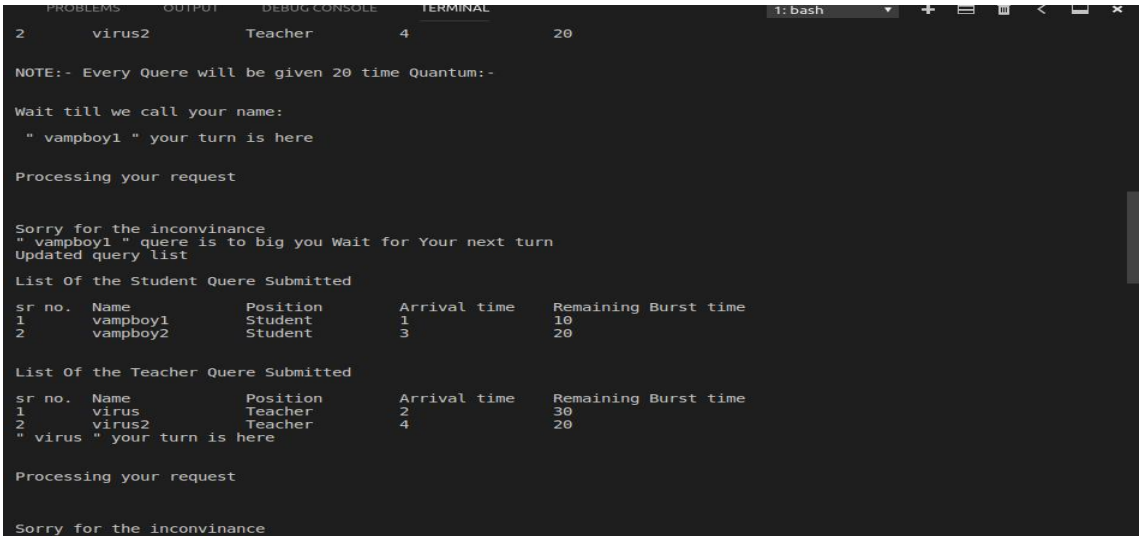
S1-pending	T1-pending	T2-completed	T1-completed	S2-completed	S1-completed
1 20	40	60	70	90	100

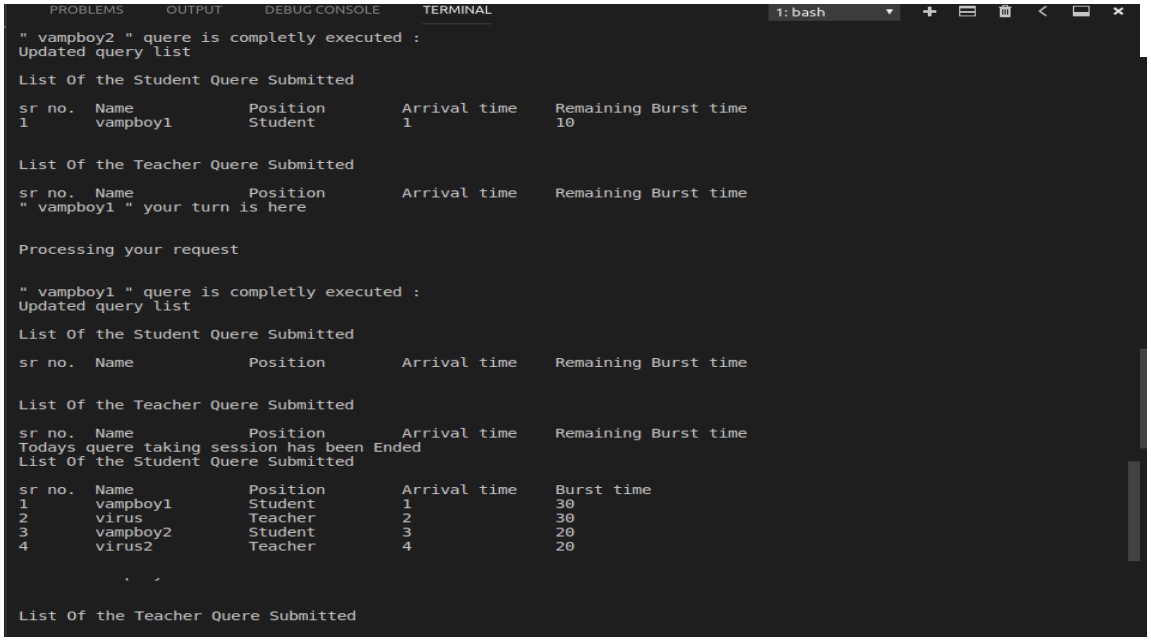
Description:- S1 will be executed first as no other has arrivied at the point the process S1 will excete for 20 timequantum and and go back to the que and wait for all the student process there execution to get his turn back. Similarly T1 will execute for 20 time quantumm and wait in que for others execution. After that T2 will be executed as it has higher priority the S2 and after the complete execution of T2 .T1 will continue its execution as all the que elements are being travarsed .after that

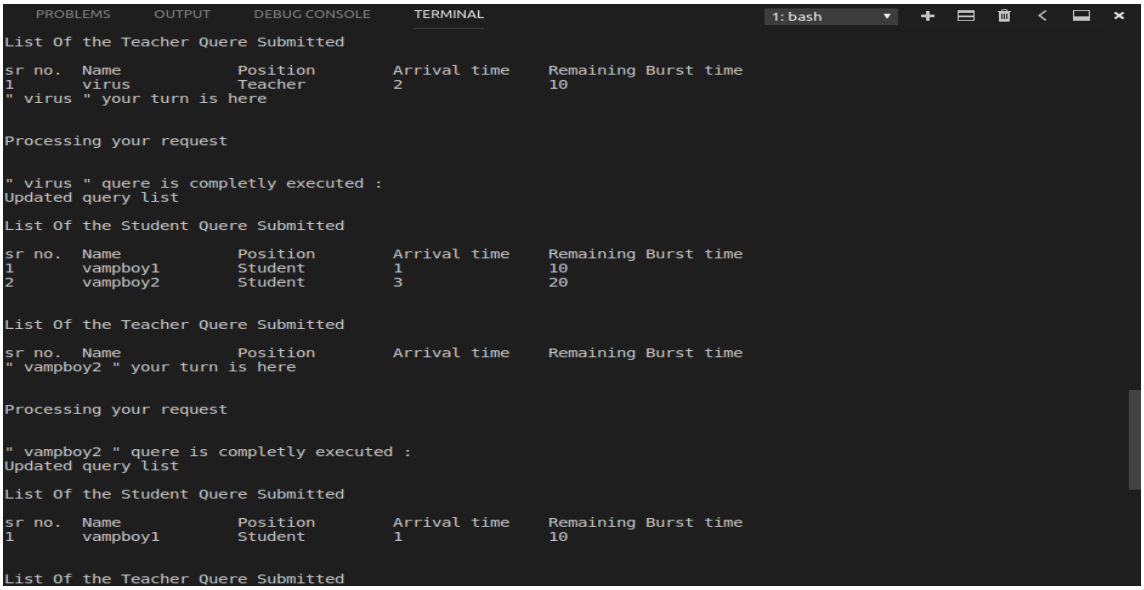


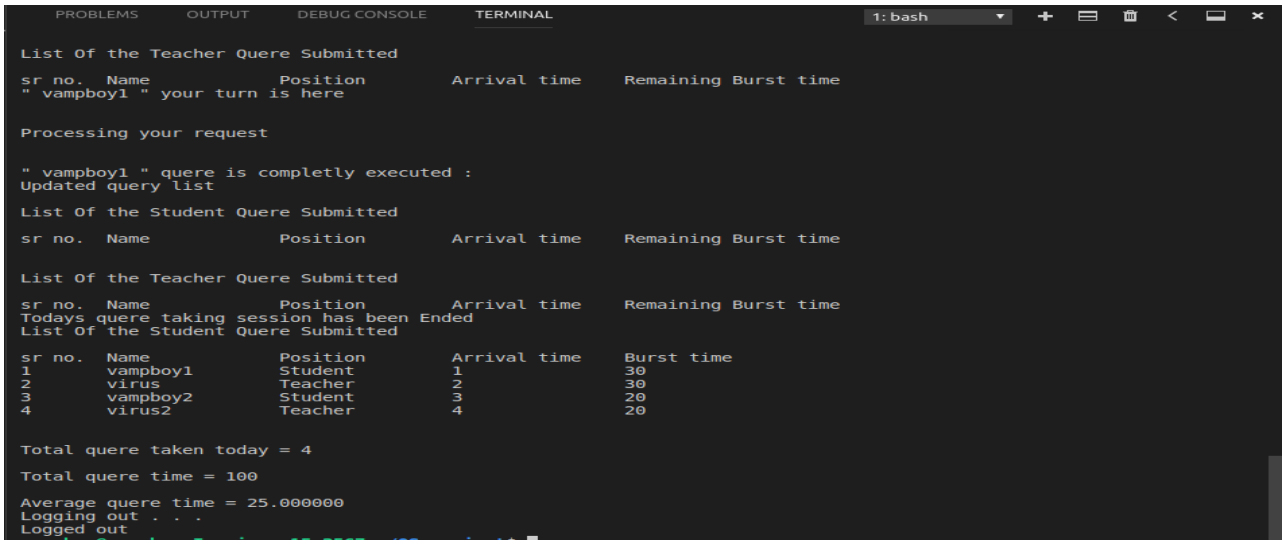
S2 will get his chance and when all the student que are traversed S1 will get his chance of traversal


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: a.out
Enter your name: vampboy2
Enter your position(student/teacher): student
Enter your arrival time: 3
Enter your quere time needed: 20
Add another form(Y/N):- y
Welcome to Quere Solutions
Enter the quere details in the form:
Enter your name: virus2
Enter your position(student/teacher): teacher
Enter your arrival time: 4
Enter your quere time needed: 20
Add another form(Y/N):- n
List Of the Student Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
1 vampboy1 Student 1 30
2 vampboy2 Student 3 20
List Of the Teacher Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
1 virus Teacher 2 30
2 virus2 Teacher 4 20


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: bash
2 virus2 Teacher 4 20
NOTE:- Every Quere will be given 20 time Quantum:-
Wait till we call your name:
" vampboy1 " your turn is here
Processing your request
Sorry for the inconvinance
" vampboy1 " quere is to big you Wait for Your next turn
Updated query list
List Of the Student Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
1 vampboy1 Student 1 10
2 vampboy2 Student 3 20
List Of the Teacher Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
1 virus Teacher 2 30
2 virus2 Teacher 4 20
" virus " your turn is here
Processing your request
Sorry for the inconvinance


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: bash
" vampboy2 " quere is completly executed :
Updated query list
List Of the Student Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
1 vampboy1 Student 1 10
List Of the Teacher Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
" vampboy1 " your turn is here
Processing your request
" vampboy1 " quere is completly executed :
Updated query list
List Of the Student Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
List Of the Teacher Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
Todays quere taking session has been Ended
List Of the Student Quere Submitted
sr no. Name Position Arrival time Burst time
1 vampboy1 Student 1 30
2 virus Teacher 2 30
3 vampboy2 Student 3 20
4 virus2 Teacher 4 20
List Of the Teacher Quere Submitted


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: bash
List Of the Teacher Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
1 virus Teacher 2 10
" virus " your turn is here
Processing your request
" virus " quere is completly executed :
Updated query list
List Of the Student Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
1 vampboy1 Student 1 10
2 vampboy2 Student 3 20
List Of the Teacher Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
" vampboy2 " your turn is here
Processing your request
" vampboy2 " quere is completly executed :
Updated query list
List Of the Student Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
1 vampboy1 Student 1 10
List Of the Teacher Quere Submitted


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1: bash
List Of the Teacher Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
" vampboy1 " your turn is here
Processing your request
" vampboy1 " quere is completly executed :
Updated query list
List Of the Student Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
List Of the Teacher Quere Submitted
sr no. Name Position Arrival time Remaining Burst time
Todays quere taking session has been Ended
List Of the Student Quere Submitted
sr no. Name Position Arrival time Burst time
1 vampboy1 Student 1 30
2 virus Teacher 2 30
3 vampboy2 Student 3 20
4 virus2 Teacher 4 20
Total quere taken today = 4
Total quere time = 100
Average quere time = 25.000000
Logging out . . .
Logged out