Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

Solution code of my project –

```
#include<stdio.h>
/*
p - priority
a,af - arrival time
b,fb - burst time
x - waiting time
tat- turnaround time
pid- process id
comp - completeion time
n -  no of process
quantum - quantum time
*/
int n,quantum;
// using structure for data
struct process_times{
   int pid,p,af,a,fb,b,comp,tat;
};
//sorting according to the arrival time and priority
void sort(struct process_times temp[],int n)
{
          int i=0,j=0,temp1=0;
          for (i = 0; i < n; ++i)
   {
      for (j = i + 1; j < n; ++j)
      {
         if (temp[i].af>temp[j].af)
         {
```

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

```
        temp1 =  temp[i].comp;
          temp[i].comp = temp[j].comp;
          temp[j].comp = temp1;
          temp1 =  temp[i].a;
          temp[i].a = temp[j].a;
          temp[j].a = temp1;
          temp1 =  temp[i].af;
          temp[i].af = temp[j].af;
          temp[j].af = temp1;
          temp1 =  temp[i].p;
          temp[i].p = temp[j].p;
          temp[j].p = temp1;
          temp1 =  temp[i].pid;
          temp[i].pid = temp[j].pid;
          temp[j].pid = temp1;
          temp1 =  temp[i].b;
          temp[i].b = temp[j].b;
          temp[j].b = temp1;
          temp1 =  temp[i].fb;
          temp[i].fb = temp[j].fb;
          temp[j].fb = temp1;
           }
                        else if(temp[i].af==temp[j].af)
                        {
                  if(temp[i].p<temp[j].p)
                                  {
          temp1 =  temp[i].comp;
          temp[i].comp = temp[j].comp;
```

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

```
            temp[j].comp = temp1;
            temp1 =  temp[i].a;
            temp[i].a = temp[j].a;
            temp[j].a = temp1;
            temp1 =  temp[i].p;
            temp[i].p = temp[j].p;
            temp[j].p = temp1;
            temp1 =  temp[i].af;
            temp[i].af = temp[j].af;
            temp[j].af = temp1;
            temp1 =  temp[i].pid;
            temp[i].pid = temp[j].pid;
            temp[j].pid = temp1;
            temp1 =  temp[i].b;
            temp[i].b = temp[j].b;
            temp[j].b = temp1;
            temp1 =  temp[i].fb;
            temp[i].fb = temp[j].fb;
            temp[j].fb = temp1;
                                    }
                            }
      }
         }
}
//sorting according to process id for final answer
void sortid(struct process_times temp[],int n)
{
          int i=0,j=0,temp1=0;
```

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

```
        for (i = 0; i < n; ++i)
  {
    for (j = i + 1; j < n; ++j)
    {
      if (temp[i].pid>temp[j].pid)
      {
       temp1 =  temp[i].comp;
        temp[i].comp = temp[j].comp;
        temp[j].comp = temp1;
        temp1 =  temp[i].a;
        temp[i].a = temp[j].a;
        temp[j].a = temp1;
        temp1 =  temp[i].af;
        temp[i].af = temp[j].af;
        temp[j].af = temp1;
        temp1 =  temp[i].p;
        temp[i].p = temp[j].p;
        temp[j].p = temp1;
        temp1 =  temp[i].pid;
        temp[i].pid = temp[j].pid;
        temp[j].pid = temp1;
        temp1 =  temp[i].b;
        temp[i].b = temp[j].b;
        temp[j].b = temp1;
        temp1 =  temp[i].fb;
        temp[i].fb = temp[j].fb;
        temp[j].fb = temp1;
      }
```

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

```
                }
    }
}
// function for display final answer with turnaround time, waiting time ,avg_wai,avg_tat and cpu utilization
void display_answer(struct process_times temp[],int n,int time,int nw){
        printf("\n\n\t Final Solution :- \n");
        float avtat=0,avwt=0;
        printf(" PROCESS ID \t| TurnAroundTime \t| Waiting Time  \n");
        for(int i=0;i<n;i++)
        {
                temp[i].tat = temp[i].comp-temp[i].a;
                int x = temp[i].comp-(temp[i].a+temp[i].b);
                if(temp[i].tat >= 0 && x>=0){
                        printf(" P[%d] \t\t| %d \t\t\t| %d \n",temp[i].pid,temp[i].tat,x);
                        avtat+=temp[i].tat;
                        avwt+=x;
                }
        }
        printf("%d %d",nw,time);
        float cpu_util = ((float)nw/(float)time) * 100;
        printf("\n\n Average TurnAround Time =  %.2f",avtat/=(n));
        printf("\n Average Waiting Time =  %.2f",avwt/=(n));
        printf("\n CPU_UTIL = %0.2f percent \n",cpu_util);
}
// main sol for gannt chart and executing the process according the algorithm
void main_sol(struct process_times temp[],int n){
        int slot = 0,time = 0,cur = 0,nw = 0;
```

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

```
        for(int i=0;i<n;i++)  //finding slots for run the actual process or distributing the process accoding to time slot
        {
                nw += temp[i].b ;
                if(temp[i].pid < n-1 && nw < temp[i+1].a){
                        slot = slot+1;
                }
                if(temp[i].b%quantum==0){
                        slot = slot+temp[i].b/quantum;
                }
                else{
                        slot = slot+(temp[i].b/quantum)+1;
                }
                // printf("%d\n",slot);
        }

        for(int i=0;i<slot;i++)
        {
                int k = 0;
                int flag =0;
                int ft=time;
                if(temp[cur].af<=time){ // in this we are updating everytime all the value
                        if(temp[cur].fb<=quantum) //running process if it equals or less than time quantum and complets it
                        {
                                time=time+temp[cur].fb;
                                temp[cur].af=temp[cur].af+temp[cur].fb;
                                k = temp[cur].fb - quantum;
                                temp[cur].fb=0;
```

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

```
                temp[cur].p=-100;
                temp[cur].af=100000;
                temp[cur].comp=time;
            }
            else  //if not equals to that
            {
                temp[cur].fb=temp[cur].fb-quantum;
                time+=quantum;
                temp[cur].af=temp[cur].af+quantum;
            }
                printf("Process P[%d]Executed From %d to %d\n",temp[cur].pid,ft,time);
        }
        else{ // if there is ideal case then this case runs and if this runs then values are not updated
            flag = 1;
        k  = temp[cur].a - ft;
        time+=k;
            printf("Process P_IDLE Executed From %d to %d\n",ft,time);

        }
        if(flag==0){  // if the process completes or having another process in queue ,for checking priority then updating the cur value
        //      printf(" P[%d] \t\t| %d \t\t| %d \t\t| %d \t\t %d\n",pid[cur],a[i],b[i],p[i],comp[i]);
        sort(temp,n);

    //      printf(" P[%d] \t\t| %d \t\t| %d \t\t| %d \t\t %d\n",pid[cur],a[i],b[i],p[i],comp[i]);
```

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

```
            printf("\n\n");
                cur=0;
                for(int j=1;j<n;j++)
                    if(temp[j].p>temp[cur].p&&temp[j].af<=time)
                        cur = j;
        }
    }
        sortid(temp,n);
        display_answer(temp,n,time,nw);
}
void display_question(struct process_times temp[],int n){ // this is what we inputted
        printf("\n\n\t\tQuestion :- \n\n");
        printf(" PROCESS ID \t| ARRIVAL TIME \t| BURST TIME \t| PRIORITY \n");
        for(int i=0;i<n;i++)
        {
            printf(" P[%d] \t\t| %d \t\t| %d \t\t| %d \t\t
%d\n",temp[i].pid,temp[i].a,temp[i].b,temp[i].p,temp[i].comp);
        }
        printf("\n\n");
    main_sol(temp,n);
}
void insert(){  // this is for taking all the values
    printf("Prioirty Based Round Robin Scheduling\n\n");
        printf("Enter Time Quantum :- ");
        scanf("%d",&quantum);
        printf("Enter Number of Process:- ");
        scanf("%d",&n);
    struct process_times temp[n];
        for(int i=0;i<n;i++){
```

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

```
            printf("\nEnter Arrval Time of Process %d :- ",i+1);
            temp[i].pid = i+1;
            scanf("%d",&temp[i].a);
            temp[i].af=temp[i].a;
            printf("Enter Burst Time Of Process %d :- ",i+1);
            scanf("%d",&temp[i].b);
            temp[i].fb = temp[i].b;
            printf("Enter Priority Of Process %d :- ",i+1);
            scanf("%d",&temp[i].p);
            temp[i].comp = 0;
            temp[i].tat = 0;
        }
        sort(temp,n);
  display_question(temp,n);
}
int main(){
   insert();
   return 0;
}
```

Ans 1). In the allocated problem, we have to input n,no of proccess to the cpu with their arrival times ,burst times and priority.Now, here comes the problem we have to solve the problem by using preemptive round robin algorithm with priority.It means we have the quantum time to solve within that time and priority so that we can allocate the n processes.Round Robin algorithm works in a premtive and cyclic way and give the min turnaround and min waiting time.Each process is assigned with a proroty,higher priority will be highest number in priority.In the case, we start the scheduler task according to priorities and the lowest priority holds in a ready queue to finish the process of highest priority.When highest priority finshed the execution according to time quantum then assigns a lower priority process but if there is no process available then P_IDLE came in a role.It means when cpu is at its ideal state no new process are availble for that time.This goes on in a cyclic way till the all process are not completed their execution.

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

Ans2,3).Algorithm for solution with complextity -

p – priority ,a,af - arrival time,b,fb - burst time,x - waiting time ,tat- turnaround time ,pid- process id,comp - completeion time,n - no of process,quantum - quantum time

Sort – Selection sort algorithm

void main_solution(p,af,a,fb,x,tat,pid,comp,n,quantum)

int slot = 0,time = 0,cur = 0,nw = 0

1.for(int i=0;i<n;i++) // complexity for this line is O(n)

//finding slots for run the actual process or distributing the process accoding to time slot

nw += temp[i].b

1. if(temp[i].pid < n-1 && nw < temp[i+1].a){ // complexity for this line is O(n)

slot = slot+1

2. if(temp[i].b%quantum==0){ // complexity for this line is O(log(n))

slot = slot+temp[i].b/quantum;

else // complexity for this line is O(n)

slot = slot+(temp[i].b/quantum)+1

2. for(int i=0;i<slot;i++) // complexity for this line is O(n)

int k = 0,flag =0,ft=time;

1. if(temp[cur].af<=time) // in this we are updating everytime all the value complexity for this line is O(n)

a. if(temp[cur].fb<=quantum) //running process if it equals or less than time quantum and complets it
complexity for this line is O(n)

time=time+temp[cur].fb

temp[cur].af=temp[cur].af+temp[cur].fb

k = temp[cur].fb – quantum

temp[cur].fb=0

temp[cur].p=-100

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

temp[cur].af=100000

temp[cur].comp=time

else //if not equals to that // complexity for this line is (n)

temp[cur].fb=temp[cur].fb-quantum

time+=quantum

temp[cur].af=temp[cur].af+quantum

else // if there is ideal case then this case runs and if this runs then values are not updated // complexity for this line is O(n)

flag = 1

k = temp[cur].a - ft

time+=k

b. if(flag==0) // if the process completes or having another process in queue ,for checking priority then updating the cur value

sort(temp,n) // complexity for this line is O(n^2)

cur=0

3. for(int j=1;j<n;j++) // complexity for this line is O(n)

1. if(temp[j].p>temp[cur].p&&temp[j].af<=time) // complexity for this line is O(n)

cur = j

sortid(temp,n) // complexity for this line is O(n^2)

So, from the implemented algorithm we have the average and worst caseoverall complexity is O(n^2) or O(n^burst times)

Ans 4).The given constrained in my problem is:

a) Each process is assigned a numerical priority, with a higher number indicating a higher relative priority.
Solution – by using an selction sort algorithm I sorted in that manner.

b) This task has priority 0 and is scheduled whenever the system has no other available processes to run.

Solution – Whenever there is no process than ther is idle condition

c) The length of a time quantum is 10 units.

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

a)

```
                }
                                else if(temp[i].af==temp[j].af)
                                {
                        if(temp[i].p<temp[j].p)
                                        {
                        temp1 =  temp[i].comp;
                        temp[i].comp = temp[j].comp;
                        temp[j].comp = temp1;
                        temp1 =  temp[i].a;
                        temp[i].a = temp[j].a;
                        temp[j].a = temp1;
                        temp1 =  temp[i].p;
                        temp[i].p = temp[j].p;
                        temp[j].p = temp1;
                        temp1 =  temp[i].af;
                        temp[i].af = temp[j].af;
                        temp[j].af = temp1;
                        temp1 =  temp[i].pid;
                        temp[i].pid = temp[j].pid;
                        temp[j].pid = temp1;
                        temp1 =  temp[i].b;
                        temp[i].b = temp[j].b;
                        temp[j].b = temp1;
                        temp1 =  temp[i].fb;
                        temp[i].fb = temp[j].fb;
                        temp[j].fb = temp1;
                                        }
                                }
            }
            }
}
```

b).

```
    else{ // if there is ideal case then this case runs and if this runs then values are not updated
            flag = 1;
    k  = temp[cur].a - ft;
    time+=k;
            printf("Process P_IDLE Executed From %d to %d\n",ft,time);


    }
```

c).

```
void insert(){  // this is for taking all the values
     printf("Prioirty Based Round Robin Scheduling\n\n");
          printf("Enter Time Quantum :- ");
          scanf("%d",&quantum);
```

Final Report CA

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

Ans 5). Yes, I have implemented an another algorithm in the solution for sorting based on priority and arrival time of process. For that, I have used an selection sort in which it compares with all the next values of the process .First of all, I sorted process acording to the arrival time for the starting the execution according to time and updating the queue.Then, it comes to priority once the process comes in queue we have to check the process priority for that to know which will execute first according to that .We check everytime whenever there is a multiple process in a queue.It is used when we are making gantt_chart, finding turn around time and waiting time.It works along with the implemented algorithm of round robin so that we get the optimized solution.The time complexity of selection sort is O(n^2)

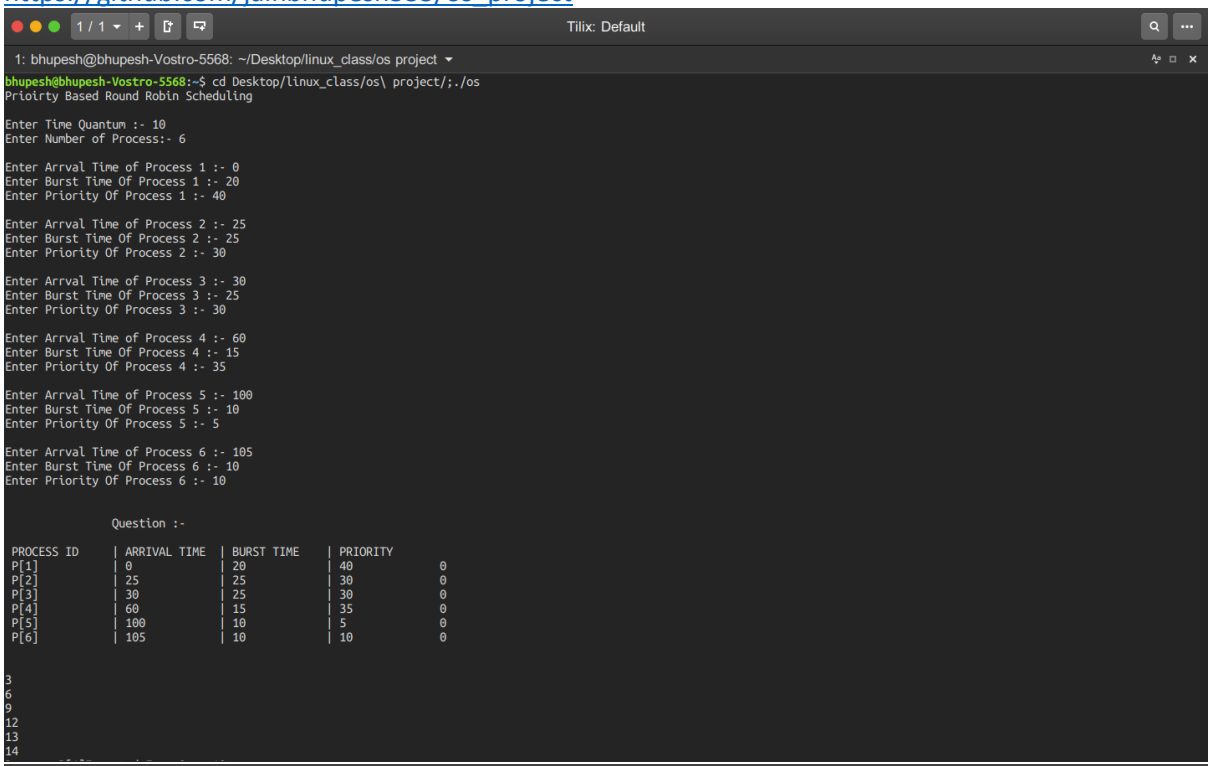
Ans 6).The boundary condition of a code is –

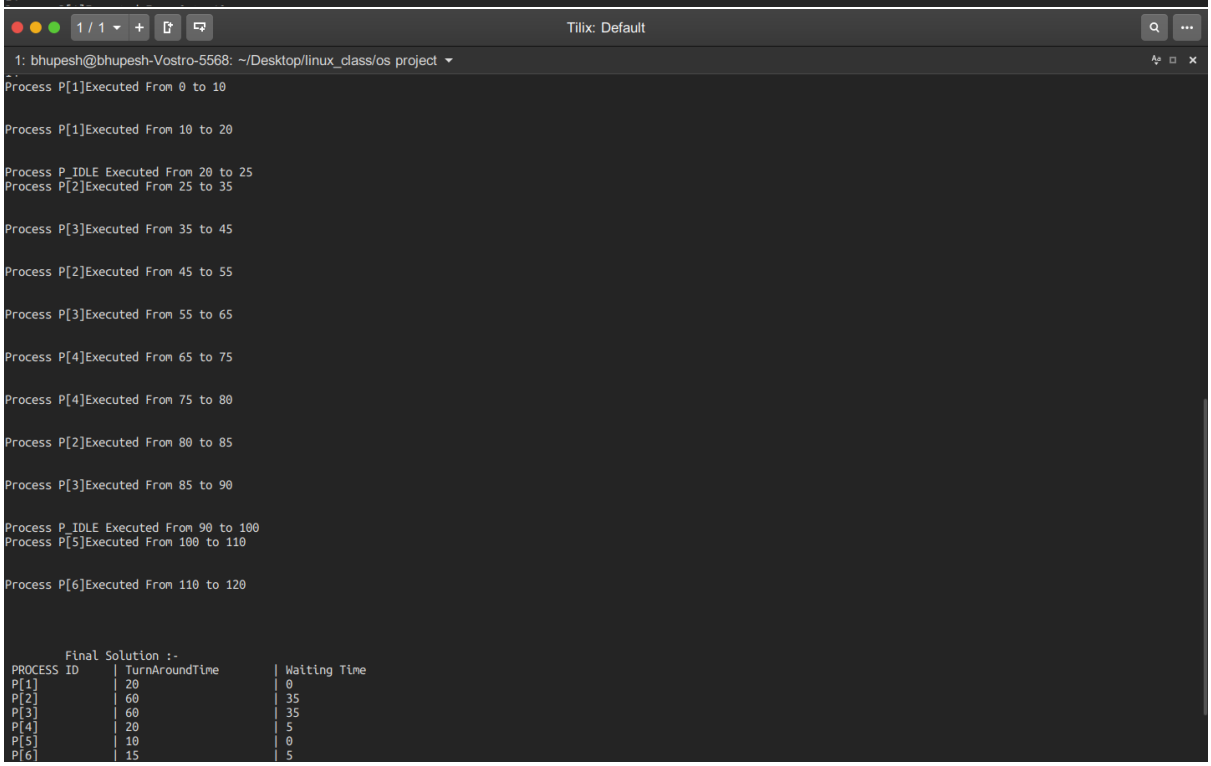
1) The value of time quantum,value of arrival time,value of burst timeand value of priority should greater than zero.
2) The value of average turn around time and average waiting time should greater than zero.
3) The value of CPU utilization should lie between in range of 0 to 100.
4) After the completion of the all the processes,all the processs data turned into negative so that repeatation of process is not allowed and code terminates the program with corect outputs.
5) Priority must be checked after every time quantum .
6) If there is no process then there should be in P_IDLE state for that time quantum.

Ans 7). I have attached two test cases, one is givern in the problem and one I have tested –

Final Report CA

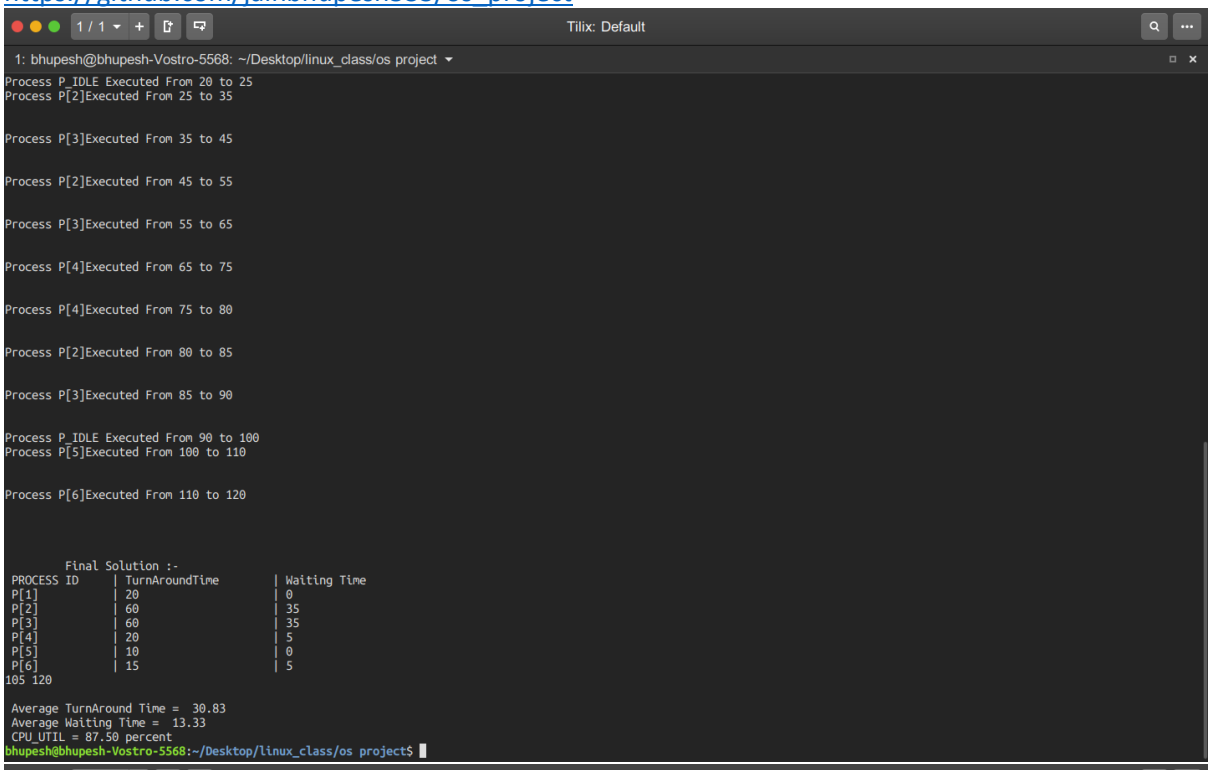
Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project

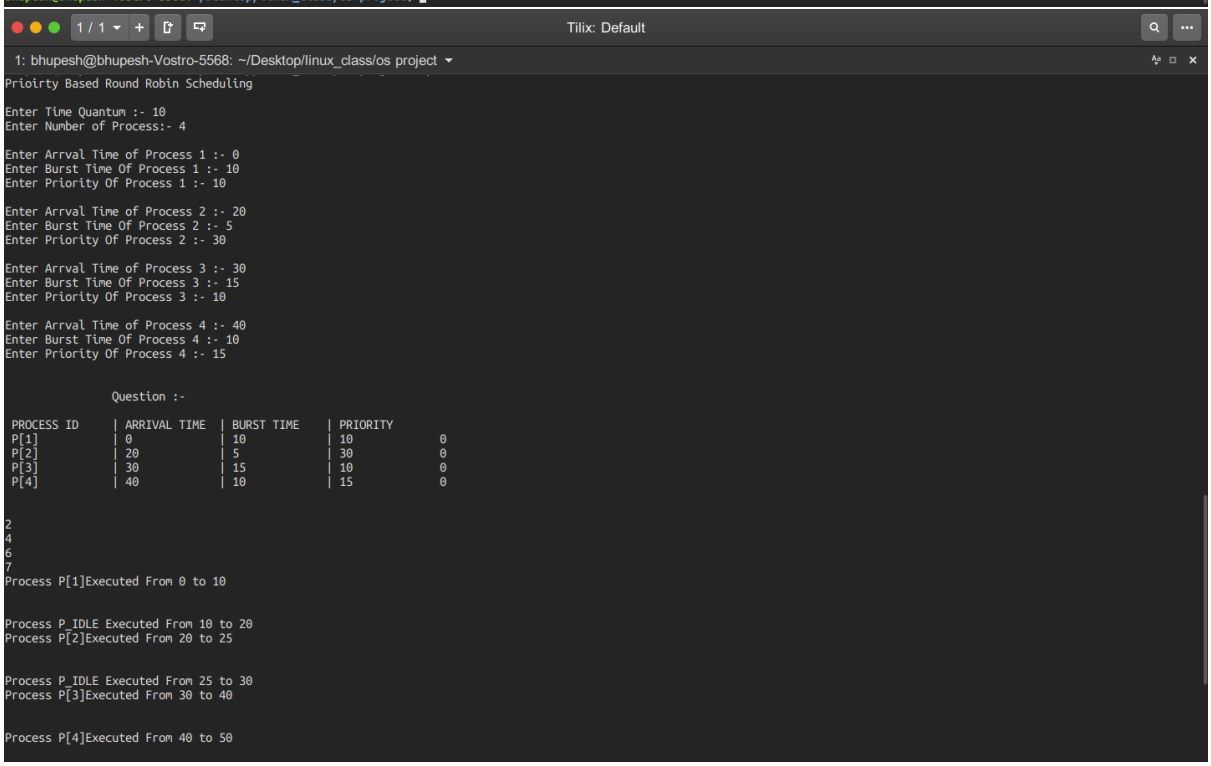




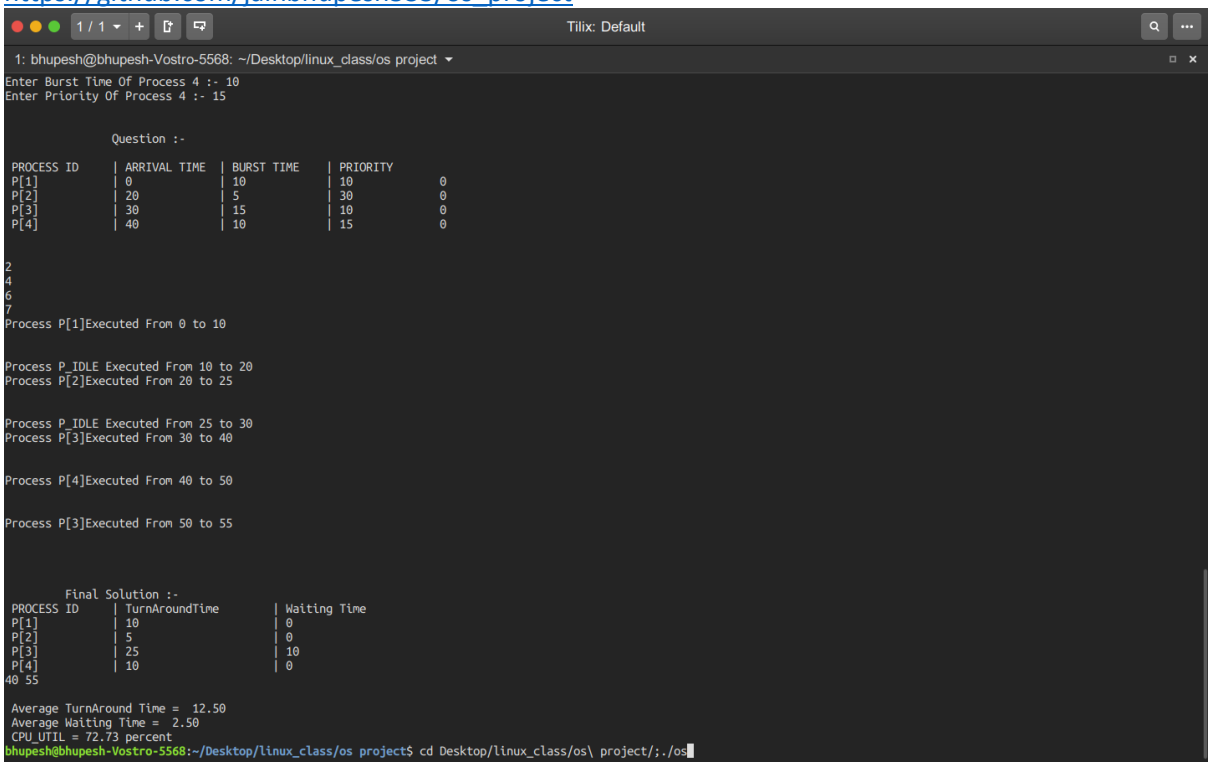
Final Report CA

Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project





Bhupesh Parakh 11803483
Jainbhupesh533@gmail.com
https://github.com/jainbhupesh533/os_project



Ans 8). Yes, I have made 5 revisions as it is uploaded on my github

Link- https://github.com/jainbhupesh533/os_project