```
//Created By Ritwik Chandra Pandey on 4/4/21
//183215
//Round Robin(Queue-LL)


#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct process {
    char name [20];
    int burst_time;
    int cpu_time;
    int wait_time;
    int rem_time;
    int turnaround_time;
};
typedef struct process* PROCESS;
struct queue {
    PROCESS proc;
    struct queue *next;
};
typedef struct queue *Q;
Q front= NULL, rear=NULL;
void enqueue (PROCESS p) {
    Q temp=NULL;
    temp=(Q)malloc(sizeof(struct queue));
    if(temp == NULL) {
        printf("Queue is overflow.\n");
    } else {
        temp -> proc=p;
        temp -> next=NULL;
        if(front == NULL) {
```

```
        front=temp;
    } else {
        rear -> next=temp;
    }
    rear = temp;    }}
PROCESS dequeue(){
    Q temp = NULL;
    PROCESS p=NULL;
    if(front== NULL) {
        printf("Queue is underflow.\n");

    } else {
        temp = front;
        if (front==rear) {
            front = rear=NULL;
        } else {
            front=front -> next;
        }
        p = temp->proc;
        free(temp);}
        return p;

}
int isEmpty(){
    if(front==NULL && rear==NULL) {
        return 1;
    }
    return 0;
}
int main(){
    int NOP, time_quantum=0,proc_time=0, current_time=0, i;
    char proc_name[20];
```

```
int sum_wait=0, sum_turnaround=0;
PROCESS p;
printf("Enter number of process : ");
scanf("%d", &NOP);
for (i=0; i < NOP; i++) {
    printf("Enter %d process name : ",i + 1);
    scanf("%s", proc_name);
    printf("Enter %d process time : ",i + 1);
          scanf("%d", &proc_time);
          p = (PROCESS)malloc(sizeof(struct process));
          strcpy(p->name, proc_name);
          p->burst_time=proc_time;
          p->cpu_time=0;
          p->wait_time=0;
          p->rem_time=proc_time;
          p->turnaround_time = 0;
          enqueue(p);
          }
printf("Enter the time quantum : ");
                scanf("%d", &time_quantum);
                printf("Printing processes in the order of their completion\n");
                printf("Proc_Name\tCPU_Time\tWait_Time\tTurnAround_time\n");
                current_time=0;
                while(isEmpty()==0){
                   p = dequeue();
                   if(p->rem_time <= time_quantum){
                      current_time+=p->rem_time;
                      p->cpu_time+=p->rem_time;
                      p->rem_time = 0;
                      p->turnaround_time=current_time;
                      p->wait_time = p->turnaround_time;
                      p->burst_time = proc_time;
```

```
            printf("%9s\t%8d\t%9d\t%15d",p->name, p->cpu_time,p->wait_time,p->turnaround_time);
            printf("\n");
            sum_wait += p->wait_time;
            sum_turnaround+=p->turnaround_time;
        }
        else{
            current_time+=time_quantum;
            p->rem_time-=time_quantum;
            p->cpu_time+=time_quantum;
            enqueue(p);
        }
    }
        printf("Total wait time : %d\n", sum_wait);
        printf("Average wait time : %.2f\n",(float)sum_wait/NOP);
        printf("Total turn around time : %d\n", sum_turnaround);
        printf("Average turn around time : %.2f\n",(float)sum_turnaround/NOP);
    }
```