```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "queue_tasks.h"
#include <time.h>
#define MAXTITLE 50
#define MaxTasks 10
#define TID_START_VALUE 101
struct Task {
  int taskID;
  char title[MAXTITLE];
  float duration;
  int status; //0 if unscheduled, 1 if scheduled, 2 if completed
};
void createFromFile(struct Task [],int);
void displayScheduledTasks(int [],int,int,struct Task *);

void delay(float number_of_seconds) //taken from gfg
{
    int milli_seconds = 1000 * number_of_seconds;
    clock_t start_time = clock();
    while (clock() < start_time + milli_seconds)
        ;
}
int main(){
  struct Task TasksQueue[10];
  int f = -1, r = -1,v,i;
  int q_of_nos[6];
  createFromFile(TasksQueue,TID_START_VALUE);
//menu driven task manager
  int c, tid,ind,topTID;
  float minw = 0,maxw = 0,w;
  do {
```

```
   printf("\nEnter:");
   printf("\n 1 to add task to scheduler");
   printf("\n 2 to run top-priority task");
   printf("\n 3 to display tasks in queue");
   printf("\n 4 to exit\n :");
   scanf("%d", &c);
   switch(c){
     case 1:
      //handle task queueing
      printf("Enter TaskID of task to be scheduled:");
      scanf("%d",&tid);
      if (((tid-100) > MaxTasks )||(tid<=100)){
        printf("Task not found ! Cannot be scheduled\n");
        continue;
      }
      if (isFull(f,r)){
        printf("Tasks full: TOO MUCH MULTITASKING IS UNHEALTHY\n");
        printf("Will have to wait between %.3f seconds to %.3f seconds for empty
slot\n", minw, maxw);
        delay(1);
        continue;
      }
      ind = tid - 101; //index of the task
      if (TasksQueue[ind].status == 0){
        printf("Scheduling task : %s\n",TasksQueue[ind].title);
        TasksQueue[ind].status = 1;
        enqueue(q_of_nos,&f,&r,tid);
        w = TasksQueue[ind].duration;
        maxw = (maxw > w)?maxw:w;
        minw = (minw < w)?minw:w;
      }else if (TasksQueue[ind].status == 1){
        printf("Task already in schedule\n");
      }else if (TasksQueue[ind].status == 2){
```

```
        printf("Task already completed\n");
       }
       break;
      case 2:
       //handle task running
       topTID = dequeue(q_of_nos,&f,&r);
       if (topTID == -1){
        printf("Empty schedule: No tasks to run\n");
        continue;
       }
       i = (topTID-101);
       printf("\nTID:%d\nTASK:%s",TasksQueue[i].taskID,TasksQueue[i].title);
       printf("\nDURATION:%.3f\n",TasksQueue[i].duration);
       //time delay for n seconds
       printf("Task in progess...\n");
       delay(TasksQueue[i].duration);
       TasksQueue[i].status = 2; //completed
       printf("\nSTATUS:Completed\n");
       break;
      case 3:
       //display current tasklist
       displayScheduledTasks(q_of_nos,f,r,TasksQueue);
       break;
      default:
       printf("Enter correct values\n");
    }
  }while(c!=4);
  return 0;
}

void createFromFile(struct Task a[], int TID_start){
  FILE * fp = fopen("input.txt","r");
  /*
```

```
  first line contains number of inputs
  first line of input contains the task string
  second line contains float duration
  */
  int n,c,e,y;
  fscanf(fp,"%d",&n);
  struct Task * q;
  char s[MAXTITLE];
  printf("Input from file\n");
  for (q = a, y = 0; y < n; y++,q++){
    fscanf(fp,"%*[\n]s");
    fgets(s,MAXTITLE,fp);
    s[strlen(s)-1] = '\0';
    strcpy(q->title,s);
    fscanf(fp,"%f",&q->duration );
    q->status = 0;
    q->taskID = TID_start++;
    printf("\nTID:%d\nTASK:%s",q->taskID,q->title);
    printf("\nDURATION:%f",q->duration);
    printf("\nSTATUS:%s\n",(q->status == 0)?"NOT DONE":"DONE");
    delay(0.3);
  }
}
void displayScheduledTasks(int b[],int front, int rear,struct Task * a){
  int f = front,r = rear;
  if (isEmpty(f, r)){
    printf("Empty tasklist\n");
    return;
  }
  struct Task * q = a;
  int i = f, j = r, k = 1,ind;
  printf("Scheduled tasks!!!\n----------\n");
  if (i > j){
```

```
    while (i < MAX){
      ind = b[i];
      q = a + (i-101);
      printf("\nTID:%d\nTASK:%s",q->taskID,q->title);
      printf("\nDURATION:%f",q->duration);
      printf("\nSTATUS:Scheduled\n");
      i++;
    }
    i = 0;
  }
  while (i <= j){
    ind = b[i];
    q = a + (ind-101);
    printf("\nTID:%d\nTASK:%s",q->taskID,q->title);
    printf("\nDURATION:%f",q->duration);
    printf("\nSTATUS:Scheduled\n");
    i++;
  }
}
```