```
/* Containes functions for HW4.c file */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <fcntl.h>
#include "queue.h"


job create_job(char *command, int job_id)
{
    job job_details;
    job_details.job_id = job_id;
    job_details.command = malloc(sizeof(char) * strlen(command));
    strcpy(job_details.command, remove_null(command));
    job_details.stat = "waiting";
    job_details.exit_stat = -1;
    job_details.start = NULL;
    job_details.stop = NULL;
    sprintf(job_details.output_file, "%d.out", job_details.job_id);
    sprintf(job_details.err_file, "%d.err", job_details.job_id);
    return job_details;
}

char *get_command(char *line){
    const char s[2] = " ";
    char *token, *temp, *x;
    temp = malloc(sizeof(char) * strlen(s));
    x = malloc(sizeof(char) * strlen(line));
    strcpy(x, line);

    token = strtok(x, s);
    token = strtok(NULL, s);
    while ((temp=strtok(NULL, s))!=NULL){
        sprintf(token, "%s %s", token, temp);
    }
    return token;
}

void show_jobs(job *jobs, int n)
{
    int i;
    if (jobs != NULL && n != 0)
    {
        printf("%-3s %-10s %10s\n", "Job ID", "Command", "Status");
        for (i = 0; i < n; ++i)
        {
            if (strcmp(jobs[i].stat, "complete") != 0)
                printf("%-3d   %-10s %10s\n", jobs[i].job_id, jobs[i].command, jobs[i].st
at);
        }
    }
}

void submit_history(job *jobs, int n)
{
    int i;
    if (jobs != NULL && n != 0)
    {
        printf("%s   |   %*s  |  %*s  |  %*s  |  %*s\n", "Job ID", -12, "Command", -20, "
Start Time", -20, "Stop Time", -6, "Status");
        for (i = 0; i < n; ++i)
        {
            if (strcmp(jobs[i].stat, "complete") == 0)
                printf("%*d   |   %*s  |  %*s  |  %*s  |  %*d\n", 3,  jobs[i].job_id, -3,
jobs[i].command, 3, jobs[i].start, 3, jobs[i].stop, -6, jobs[i].exit_stat);
        }
    }
}
```

```
queue *queue_init(int n)
{
    queue *job_queue = malloc(sizeof(queue));
    job_queue->size = n;
    job_queue->buffer = malloc(sizeof(job *) * n);
    job_queue->start = 0;
    job_queue->end = 0;
    job_queue->count = 0;

    return job_queue;
}

int queue_insert(queue *job_queue, job *current_jobs)
{
    if ((job_queue == NULL) || (job_queue->count == job_queue->size))
        return -1;

    job_queue->buffer[job_queue->end % job_queue->size] = current_jobs;
    job_queue->end = (job_queue->end + 1) % job_queue->size;
    ++job_queue->count;

    return job_queue->count;
}

job *queue_delete(queue *job_queue)
{
    if ((job_queue == NULL) || (job_queue->count == 0))
        return (job *)-1;

    job *job_details = job_queue->buffer[job_queue->start];
    job_queue->start = (job_queue->start + 1) % job_queue->size;
    job_queue->count = job_queue->count -1;

    return job_details;
}

void queue_destroy(queue *job_queue)
{
    free(job_queue->buffer);
    free(job_queue);
}

char *current_time()
{
    int i, j;
    time_t curr_time = time(NULL);
    char *time_str;
    time_str = malloc(sizeof(char) * strlen(ctime(&curr_time)));
    strcpy(time_str, ctime(&curr_time));
    i = -1;

    while ((j = time_str[++i]) != '\0' && j != '\n')
        time_str[i] = j;
    time_str[i] = '\0';

    return time_str;
}


char **get_args(char *line)
{
    char *copy = malloc(sizeof(char) * (strlen(line) + 1));
    strcpy(copy, line);

    char *arg;
    char **args = malloc(sizeof(char *));
    int i = 0;
    while ((arg = strtok(copy, " \t")) != NULL)
```

```
    {
        args[i] = malloc(sizeof(char) * (strlen(arg) + 1));
        strcpy(args[i], arg);
        args = realloc(args, sizeof(char *) * (++i + 1));
        copy = NULL;
    }
    args[i] = NULL;
    return args;
}


int file_open(char *fn)
{
    int fd;
    if ((fd = open(fn, O_CREAT | O_APPEND | O_WRONLY, 0755)) == -1)
    {
        fprintf(stderr, "Error: failed to open \"%s\"\n", fn);
        perror("open");
        exit(-1);
    }
    return fd;
}

char *remove_null( char *s )
{
    s[strcspn ( s, "\n" )] = '\0';
    return s;
}
```