TOBB University of Economics and Technology
Department of Computer Engineering
BIL395 Programming Languages
Instructor: Dr. Osman Abul

Assignment 2[1]
Date due: March 2, 2018

Subject: C programming

Problem: Building and simulating a *Least-Slack-Time* scheduler using linked list and stack data structures

Description: In time shared systems, tasks submitted to the system are assigned to a limited number of processors for execution in a time shared manner. Time allocation of processors to tasks is known as *scheduling* problem. One of the most prevalent scheduling policy employed in embedded systems is *Least-Slack-Time first*. The policy always executes the task having the least runtime left. In this assignment, you are expected to write a C console program simulating a *Least-Slack-Time first* scheduler on a single processor system. The functionalities and constraints are specified below.

- Each task has a task *identifier* which is a string. You may assume that no two tasks are given the same identifier. Additionally, each task is assigned with a positive integer denoting the *runtime* (the total time the task must spend on the processor) of the respective task.
- There is a (waiting) *task list* keeping all the tasks submitted to the system but not completed yet, *i.e.*, partially completed tasks.
- At a time, the scheduler picks a task (according to the policy) from the task list and executes it for 1 unit time. This step decrements the runtime of the task by 1 unit. In case the runtime reaches 0, the task is added to the *completed tasks stack*. Otherwise, the task is reinserted to the task list. This process is called a simulation step and repeated.
- New tasks can be submitted to the system at any time.
- An entry in the *completed tasks stack* keeps the respective task identifier, its initial runtime and completion time (the number of time units

[1]Submissions should be sent to the assistant by e-mail.

elapsed since the submission till the completion). Note that, the completion time is initial runtime plus the time spent in the task list.

- The system time is 0 at the beginning, and increases by 1 unit at each simulation step.
- Whenever the task list becomes empty, the processor goes on *idle*. Assume that as the processor goes on idle, the system time progresses 1 unit only, regardless of the actual real time spent during which.

Program: Your program should accept commands from the command prompt in a loop.

Commands: The format is as follows:

- Display the information of most recently completed k tasks (the most recent first): > `MRCT` ***k***
- Empty completed tasks stack: > `ECTS`
- Dump (but not delete) completed tasks stack content into a text file (the most recent first): > `DFCTS` *file_name*
- Delete k earliest completed tasks from the stack: > `DECTCTS` ***k***
- Display system time: > `DST`
- Display total (*idle*) time: > `DIT`
- Start a new task : > `SNT` *task_name runtime*
- Simulate k units : > `S` ***k***
- Quit the program: > `QUIT`

Important:

- Use only C constructs, do not use C++. So, for instance, instead of **class**, use **struct**.
- Never use ready-made data structures from any library.
- Use linked list data structure for implementing the task list. You are expected to use pointers. Hence, array based implementations will be awarded marginally.
- You are allowed to use arrays for stack implementation.