EE306 Introduction to Computing

Lab 6 (due on 11/13, 9pm, on GitHub)
Course Instructor: Dr. Nina Telang

All Lab assignments must be completed individually. You are not permitted to seek help or clarification from anyone other than the instructor or the TAs.

Your file should be named exactly after your EID, for example, xy1234.asm. Your program will not be graded if you fail to follow the naming convention of the file name.

Purpose: The purpose of this assignment is to write a program in **LC-3 assembly language** that creates a user interface to move students in the waiting room of the EE 306 Zoom classroom to the main room**.** The list of participants is organized as a "**LINKED LIST**" data structure. There are 2 linked lists: one which contains the EIDs of the students already in the main classroom and the second list contains the EIDs of students currently in the waiting room.

Note that you can use your program for Lab5 as a starting point for this lab.

Your program must:

1. Prompt the user for the student's EID **(refer to EID description in point 6)** by printing on the monitor the string "**Type EID and press Enter:** " and wait for the user to input a string followed by <Enter> (ASCII: x0A). **(Assume that there is no case where the user input exceeds or is less than the required number of characters)**
2. Your program must then search the **main room student list** to find a match for the entered EID. The list stores the student EID for each student. You will find a match only if the student's EID is in the list. It is possible to not find a match in the list.
3. If your program finds a match, then it must print out "**<EID> is already in the main room.". (eg., "XY123 is already in the main room.")**
4. If a match is not found in the main room student list, then your program should **check the second list of students in the waiting room** for the user entered EID. If there is a match in second list, remove the node from the second list and add it to the front of the main room list and print out "**<EID> is added to the main room.". (eg., "XY123 is added to the main room.")**
5. If the entered EID does not find a match in the second list as well, then you must print out the string, "**The entered EID does not match.**".
6. **Unique EIDs** are **exactly 5 characters long** and contains only uppercase alphabets and numbers**.** For example, XY123. **Note: first two characters must be uppercase letters and last 3 characters must be numbers.**
7. Head pointer for the main room & waiting room lists are at x4000 and x4001 respectively. In other words, the address of the first nodes of the two lists are at **x4000 and x4001** respectively.

The Linked-List

A linked-list is a set of nodes connected to each other via pointers. Each node in the linked-list contains a pointer to (the address of) the next node in the linked-list. This pointer is commonly known as the next-pointer. If the last node contains x0000 as its next pointer, it implies that there is no "next node." That is, this is the last node. We call x0000 in this context a NULL pointer.

Each node in a linked-list is comprised of *k+1* words: one word containing the next-pointer (the pointer to the next node) and *k* words of data which are being stored by the node. In our case, the database of student IDs is implemented as a linked-list with *k+1=2*. Each node consists of two words in the following order:

1. The next-pointer.
2. A pointer to an ASCII string representing the student's unique EID (5 Characters long+ NULL).

Recall that a string consists of ASCII codes stored in consecutive memory locations, one ASCII code per location. The string is null-terminated, i.e., the end of a string is signified by the NULL character which is ASCII code 0.

Below is an example database implemented as a linked-list. When you test your program, you can use a database with a similar structure. The test cases that we will use to test your program will be similar to this example.

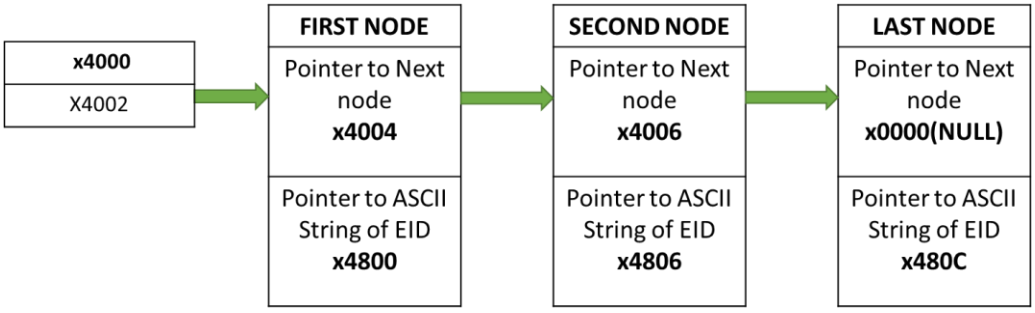

Figure 1: Example Linked List Structure – Main Room Participants List

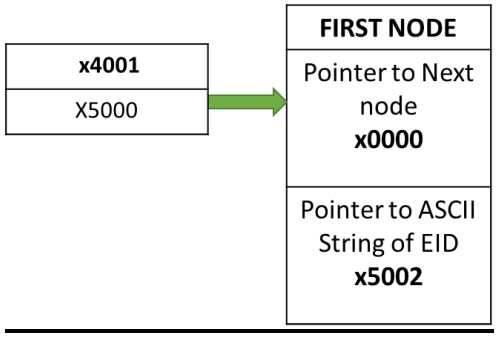

Figure 2: Example Linked List Structure – Waiting Room List

Address	M[Address]
X4000	X4002
X4001	X5000
X4002	X4004 **(Next pointer)**
X4003	X4800 **(Pointer to EID)**
X4004	X4006
X4005	X4806
X4006	X0000
X4007	X480C
…	…
X4800	"X"
X4801	"Y"
X4802	"1"
X4803	"2"
X4804	"3"
X4805	X0000
X4806	"A"
X4807	"B"
X4808	"7"
X4809	"8"
X480A	"9"
X480B	X0000
X480C	"P"
X480D	"Q"
X480E	"5"
X480F	"6"
X4810	"7"
X4811	X0000
…	…
X5000	X0000
X5001	X5002
X5002	"M"
X5003	"N"
X5004	"4"
X5005	"8"
X5006	"3"
X5007	X0000

Figure 3: Contents of Memory Before Execution

If the input were:

Type EID and press Enter: MN483

Then the contents of memory changes as follows:

Address	M[Address]
X4000	**X5000**
X4001	**X0000**
X4002	X4004 **(Next pointer)**
X4003	X4800 **(Pointer to EID)**
X4004	X4006
X4005	X4806
X4006	X0000
X4007	X480C
…	…
X4800	"X"
X4801	"Y"
X4802	"1"
X4803	"2"
X4804	"3"
X4805	X0000
X4806	"A"
X4807	"B"
X4808	"7"
X4809	"8"
X480A	"9"
X480B	X0000
X480C	"P"
X480D	"Q"
X480E	"5"
X480F	"6"
X4810	"7"
X4811	X0000
…	…
X5000	**X4002**
X5001	X5002
X5002	"M"
X5003	"N"
X5004	"4"
X5005	"8"
X5006	"3"
X5007	X0000

Figure 4: Contents of Memory After Execution

EE306 Introduction to Computing

Input/Output Requirements

Described below are detailed requirements about the Inputs and Outputs of your program. You should adhere to these guidelines to receive full credit for this assignment.

Input: Your program should prompt the user for the last name from the keyboard, as follows: Print a string EXACTLY "**Type EID and press Enter:** ". Then wait for the user to input a string followed by <Enter>. Note that you will get a 0 on the assignment if you do not print this string EXACTLY. The user will input a character string from the keyboard, terminating the EID with the <Enter> key.

Hint: To continually read from the keyboard without first printing a prompt on the screen, use TRAP x20 (assembler name GETC). That is, for each key you wish to read, the LC-3 operating system must execute the TRAP x20 service routine. If you follow TRAP x20 with the instruction TRAP x21 (assembler name OUT), the character the user types will be displayed on the screen.

Output: Your program should output one of three strings depending on the outcome of the linked list lookup. When the ID entered by the user is found in the main room list, print out "**<EID> is already in the main room.**". If the EID is found in the second list, remove the node from the second list and add it to the front of the main room student list and print "**<EID> is added to the main room."**. If the entered EID does not find a match in both the lists then print out the string, "**The entered EID does not match.**".

Hint: To output a string to the console display, use TRAP x22 (assembler name PUTS). What needs to go into R0 to use this TRAP instruction?

A sample of what your program will produce, when supplied with the input from the user trying to add the course:

EXAMPLE 1:

Type EID and press Enter: MN483
MN483 is added to the main room.

----- Halting the processor -----

EXAMPLE 2:

Type EID and press Enter: HI654
The entered EID does not match.

----- Halting the processor -----

EE306 Introduction to Computing

EXAMPLE 3:

Type EID and press Enter: XY123
XY123 is already in the main room

----- Halting the processor -----

Notes & Hints:

1. Your program must start at location x3000.

2. The two linked lists representing the Zoom Classroom participant list is an input to your program. The list is loaded in memory before your program begins to run. Your program will search the list and modify it.

3. The pointer to the first node to the main room list and the waiting room list is stored in memory location x4000 and x4001 respectively before the program execution begins. Further, assume that all the nodes are stored between memory locations x4002 and xFDFF. Make no other assumptions about the location of the nodes. **Note that the pointer to the first node may be set to NULL (i.e., 0), indicating that there are no nodes in the list.**

4. Your program should NOT make any assumptions about the number of nodes in the list.

5. You may assume that everyone in the list has a unique EID.

6. The <Enter> key is the carriage return character which is the ASCII code x0A.

7. Do not forget to modify the content of memory locations x4000 and x4001 while removing and adding nodes.

8. Do not forget to echo characters while reading from the console.

9. Pay attention to changes between Figures 3 and 4.

10. You can reduce the size of your code by defining the common parts required for both directories as **subroutines.**

IMPORTANT: The file that you will upload to online repository for this assignment must be named **youreid.asm. (eg., xy123.asm)**