CMPT350: Laboratory 4

Linked Lists in Assembly *5%*

Due: December 1st, 2022 @ 17:00

Now that we've made a "node" in Lab Demo 8, we can link multiple nodes together using the last field in the node to form a list of nodes we can navigate from front to back – a linked list!

Laboratory Procedure:

On the course Moodle page, look for the Laboratory 4 assignment in the Laboratories section. Inside is a premade skeleton MIPS files, `laboratory4.s.` Save this somewhere handy on your machine. You may use both the command line version, `spim,` and the GUI version, QtSpim to run your program. As well, you will need a text editor to edit your programs.

Assignment Requirements:

- Create a loop in the main portion of your program which will accept an integer input from the user, and take one of three actions:
  - If the user provides a "1", lead the user through the process of creating a new node (I.e., what we did in Demo 8). This node should be placed at the end of the list.
    Consider: Making a function which will create the node for you and return the address of that node
  - If the user provides a "2", prompt them for an ID value, and search the linked list for their id number, and print the contents of the node.
  - If the user provides a "3", print a goodbye message and exit the program.

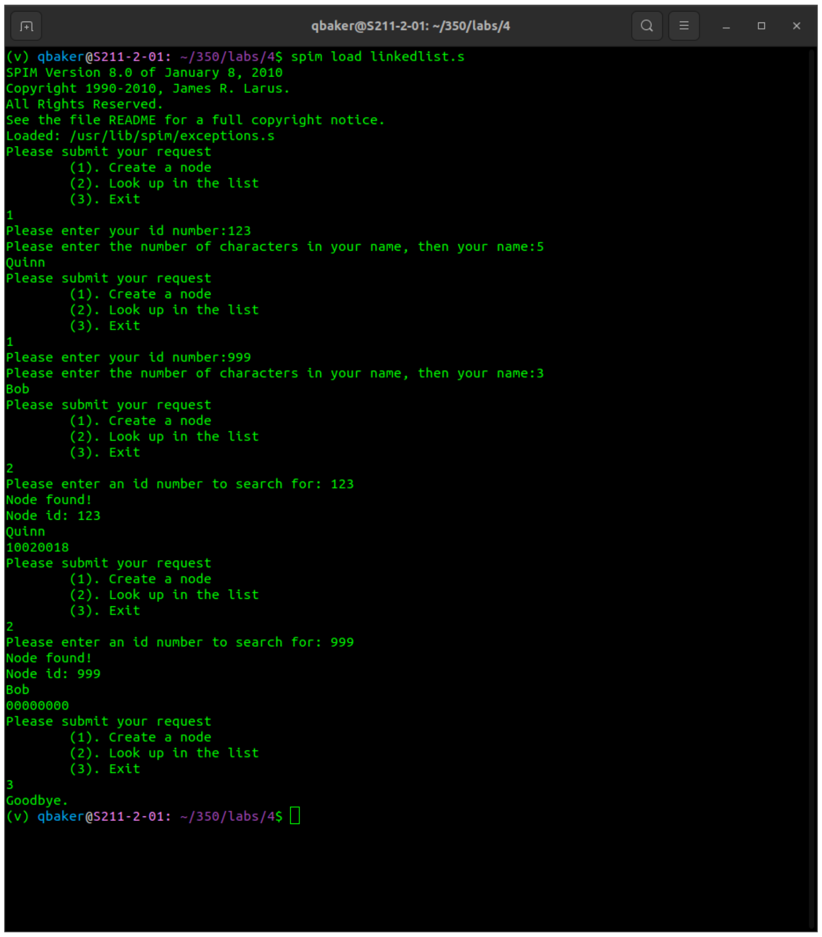
*** **NOTE:** Your code **must interpret and execute** within the **laboratory environment**. Failure to do so will result in a **mark of ZERO** for the program. ***

Submission

Your programs should be named using your name and end in a .s extension. For example, BakerLaboratory4.s

When you feel your programs meet all of the above requirements, create a zip archive with your code submission and trace file(s) included. An easy way to do this is with the `zip` command. For example, if your solution file and trace file are in a folder called `baker`, then the command `zip -r baker.zip baker/` will create a zip file containing your submission files called baker.zip (Replace this with your name when creating your zip file). Upload this zip file to Moodle.

Example Run



```
(v) qbaker@S211-2-01: ~/350/labs/4$ spim load linkedlist.s
SPIM Version 8.0 of January 8, 2010
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: /usr/lib/spim/exceptions.s
Please submit your request
        (1). Create a node
        (2). Look up in the list
        (3). Exit
1
Please enter your id number:123
Please enter the number of characters in your name, then your name:5
Quinn
Please submit your request
        (1). Create a node
        (2). Look up in the list
        (3). Exit
1
Please enter your id number:999
Please enter the number of characters in your name, then your name:3
Bob
Please submit your request
        (1). Create a node
        (2). Look up in the list
        (3). Exit
2
Please enter an id number to search for: 123
Node found!
Node id: 123
Quinn
10020018
Please submit your request
        (1). Create a node
        (2). Look up in the list
        (3). Exit
2
Please enter an id number to search for: 999
Node found!
Node id: 999
Bob
00000000
Please submit your request
        (1). Create a node
        (2). Look up in the list
        (3). Exit
3
Goodbye.
(v) qbaker@S211-2-01: ~/350/labs/4$
```

Figure 1: An example run of the Linked List program. This image is also on Moodle in the Lab 4 assignment section.

Hints

- Functions are your friends – the instructor's solution uses a function to create a node and return its address, and a function to print the contents of a node
- Keeping the address of the head node of the list in a permanent register is useful as an anchor point in the list.
- As you've no doubt learned by now, assembly programming is tedious and mind-numbing. This can be abated by ample project planning, working out your plan to solve the problem on paper (or the whiteboard), and good commenting. This project can be broken up into small manageable chunks (create the menu loop, create node, add node to list, search for a node, print a node). Completing one step, commenting it, and planning the next step should prove very fruitful