Lab 13: Generate Linked List

CSci 515 Spring 2015

2015-04-21

Dates:

Due: In Lab, Wednesday April 29, by 4:05 pm (lab end time)

Objectives

- Use dynamic memory.
- Manipulate linked lists
- Practice using pointers
- Practice processing linked list data structures and following pointers.

Description

Using the linked list data structure we developed in class today write the following function. Write a function that creates a linked list of randomly generated integers. This function will take a single parameter `numNodes` and generate that number of nodes in the linked list. Then you will display the values in the linked list.

Perform the following tasks:

1. Create a function called `generateRandomList()`. This function should take a single parameter `numNodes` as its input. It will generate a linked list with that number of nodes in it. The `Node` of the linked list will hold a single integer value. This function should generate a random integer in the range from 1 to 20 and initialize each of the nodes it creates with this random value. This function should return a `Node*` pointer as its result, which will be a pointer to the first node in the

generated linked list. The nodes created by the linked list should be generated using dynamic memory allocation. **NOTE**: you can always assume `numNodes` >= 1, thus you don't have to deal with worrying about an empty list being generated here.

2. In your main function, display the returned values from generating a list with 15 nodes in it. You should use a while loop to follow the node links, and process and display the value of each `Node` item in the list to standard output. See the example output for how your results should be formatted.

Example output is shown below:

```
List of 15 integers:
4 -> 7 -> 18 -> 16 -> 14 -> 16 -> 7 -> 13 -> 10 -> 2 -> 3 -> 8 -> 11 -> 20 -> 4 -> EOL
```

NOTE: Now that our programs have more functions than just the `main()` function, the use of the function headers becomes meaningful and required. Make sure that all of your functions have function headers preceding them that document the purpose of the functions, and the input parameters and return value of the function.

Lab Submission

An eCollege dropbox has been created for this lab. You should upload your version of the lab by the end of lab time to the eCollege dropbox named `Lab 13 Genrate Linked List`. Work submitted by the end of lab will be considered, but after the lab ends you may no longer submit work, so make sure you submit your best effort by the lab end time in order to receive credit.

Requirements and Grading Rubrics

Program Execution, Output and Functional Requirements

1. Your program must compile, run and produce some sort of output to be graded. 0 if not satisfied.

2. 40+ Your `generateRandomList()` function must work correctly and take the specified input parameters..

3. 30+ pts. Your funciton must use dynamic memory allocation to create the correct number of nodes, and set up and return the linked list correctly.

4. 30+ pts. You main function demonstrates correctly invoking the functions and displays the required output from doing so.

Program Style

Your programs must conform to the style and formatting guidelines given for this course. The following is a list of the guidelines that are required for the lab to be submitted this week.

1. The file header for the file with your name and program information and the function header for your main function must be present, and filled out correctly.

2. A function header must be present for all functions you define. You must document the purpose, input parameters and return values of all functions. Your function headers must be formatted exactly as shown in the style guidelines for the class.

3. You must indent your code correctly and have no embedded tabs in your source code. (Don't forget about the Visual Studio Format Selection command).

4. You must not have any statements that are hacks in order to keep your terminal from closing when your program exits (e.g. no calls to system()).

5. You must have a single space before and after each binary operator.

6. You must have a single blank line after the end of your declaration of variables at the top of a function, before the first code statement.

7. You must have a single blank space after , and ; operators used as a separator in lists of variables, parameters or other control structures.

8. You must have opening { and closing } for control statement blocks on their own line, indented correctly for the level of the control statement block.

9. All control statement blocks (if, for, while, etc.) must have `{ }` enclosing them, even when they are not strictly necessary (when there is only 1 statement in the block).

10. You should attempt to use meaningful variable and function names in your program, for program clarity. Of course, when required, you must name functions, parameters and variables as specified in the assignments. Variable and function names must conform to correct `camelCaseNameingConvention` .

11. Put the `*` for pointer variable declarations next to the type declaration, with no space between the type and the `*`. Also please follow the convention of using `Ptr` at the end of names for pointer variables.

Failure to conform to any of these formatting and programming practice guidelines for this lab will result in at least 1/3 of the points (33) for the assignment being removed for each guideline that is not followed (up to 3 before getting a 0 for the assignment). Failure to follow other class/textbook programming guidelines may result in a loss of points, especially for those programming practices given in our Deitel textbook that have been in our required reading so far.