CptS 122 – Data Structures

Lab 5: More Practice with Pointers, Lists, Stacks, and Queues in C (Last Lab in C!)

Assigned: Wednesday, June 21, 2017
Due: At the end of the lab session

I. Learner Objectives:

At the conclusion of this programming assignment, participants should be able to:

- Compare and contrast dynamic linked lists, dynamic stacks, and dynamic queues
- Distinguish between various types of pointers
- Dive deeper into the <string.h> library and describe the purpose of `memset()`, `memcpy()`, `memcmp()`, and `memchr()`

II. Prerequisites:

Before starting this programming assignment, participants should be able to:

- Analyze a basic set of requirements for a problem
- Compose a small C language program
- Create test cases for a program
- Apply and implement structs in C
- Apply and implement pointers in C
- Apply and implement dynamic memory in C
- Design and implement a dynamic singly linked list
- Design and implement a dynamic linked stack
- Design and implement a dynamic linked queue

III. Overview & Requirements:

This lab will allow you to further explore pointers and memory, linked lists, stacks, and queues.

Labs are held in a “closed” environment such that you may ask your TA questions. Please use your TAs knowledge to your advantage. You are required to move at the pace set forth by your TA. Please help other students in need when you are finished with a task. You may work in pairs if you wish. However, I encourage you to compose your own solution to each problem. Have a great time! Labs are a vital part to your education in CptS 122 so work diligently.

Tasks:

1. Given the following fragment of C code, answer the provided questions. If necessary, you could copy and paste the code into Visual Studio to answer the questions.

```
Line 1: int n1 = 10, n2 = 42, list[] = {6, 8, 42, 3, 2, 2, -6};

Line 2: int * const p1 = &n1;
Line 3: const int * p2 = &n1;
      Line 4: int * p3 = list;
```

```
        Line 5: const int * const p4 = NULL;

        Line 6: *p1 = 15;
Line 7: p1 = &n2;

Line 8: p2 = &n2;
    Line 9: *p2 = 67;

    Line 10: p3[4] = 67;

Line 11: list = &n1;

        Line 12: p4 = list;
    Line 13: *p4 = 25;
```

a. Is there any discernible difference between the declared types for `p1` and `p2` on lines 2 and 3?
b. Is the assignment operation on line 6 legal?
c. Is the assignment operation on line 7 legal?
d. Is the assignment operation on line 8 legal?
e. Is the assignment operation on line 9 legal?
f. Is the assignment operation on line 10 legal?
g. Is the assignment operation on line 11 legal?
h. Is the assignment operation on line 12 legal?
i. Is the assignment operation on line 13 legal?

2. Complete the implementation for the following function. The function should allocate space on the heap for the given string parameter, and copy (use `strncpy()`) the string (`pStr`), including the null character to the heap. The function should return NULL if memory could not be allocated; otherwise it should return a pointer to the beginning of the allocated block of memory on the heap.

```
char * copyStrToHeap(char *pStr)
{







}
```

3. Examine the following standard library functions, which are located in <string.h>. As you examine each one at the provided URL, please run the code examples in the C++ shell (click on the gear next to the example code).
   a. memset () - http://www.cplusplus.com/reference/cstring/memset/
   b. memcpy () - http://www.cplusplus.com/reference/cstring/memcpy/
   c. memcmp () - http://www.cplusplus.com/reference/cstring/memcmp/
   d. memchr () - http://www.cplusplus.com/reference/cstring/memchr/

In teams discuss the purpose of each function. Could you apply each one, independent of the provided code examples? Could you replace the `strncpy()` operation in Task 2 with a `memcpy()`? Could you use `memcpy()` to copy structs?

For the following problems define a `struct node` with data of type `int`.

4. Implement a function that merges two linked lists in order. The function must be able to merge two ordered lists in descending or ascending order. The function accepts pointers to the two lists and an order to merge the lists, and returns a pointer to the beginning of the merged lists. Be sure to test your function! You may start with code that you have written for other labs. EXTRA CHALLENGE: Could you implement a merge lists function which merges `n` lists in order? This function would accept a pointer to an array of `n` pointers, where each pointer refers to a list.

5. Implement a function that detects if there is a “loop” in a singly linked list. A “loop” is defined as a link from one node to itself or to another node that is before the node in the list. The function may only make one pass through the list. It must return 1 if a “loop” exists; 0 otherwise. Be sure to test your function! You may start with code that you have written for other labs. **This is a common interview question!**

6. Implement a function that reverses a singly linked list in one pass of the list. The function accepts a pointer to the beginning of the list, and returns a pointer to the beginning of the reversed list. The function must not create any new nodes. Be sure to test your function! You may start with code that you have written for other labs. **This is a common interview question!**

7. This problem does not require that you implement any C code. On the whiteboards in the lab, you and two other students must write pseudocode for the following problem:

    Write a pseudocode algorithm that uses two stacks to simulate one queue. Only consider insertions and deletions from the queue. Since this is pseudocode, you do not have to describe how to implement push and pop (these are the operations that you will discuss to simulate the queue). However, you may assume that these are acceptable terms to use within your pseudocode algorithm. Make sure that the algorithm clearly describes what needs to be done with the stack during insertions and deletions from the queue.

8. Implement a list using an array, instead of a linked implementation. With your team members discuss the following questions:
    a. Is it more efficient to delete the last node in an array or linked implementation of a list?
    b. Is it more efficient to delete the first node in an array or linked implementation of a list?
    c. Is it more efficient to delete a node, in general, in an array or linked implementation of a list?
    d. Is it more efficient to insert a node at the end in an array or linked implementation of a list?
    e. Is it more efficient to insert a node at the front in an array or linked implementation of a list?
    f. Is it more efficient to insert a node, in general, in an array or linked implementation of a list?
    g. Is it more efficient to access a node, in general, in an array or linked implementation of a list?

9. Work on programming assignment 3

IV. Submitting Labs:

- You are not required to submit your lab solutions. However, you should keep them in a folder that you may continue to access throughout the semester. You should not store your solutions to the local C: drive on the EME 120/128 machines. These files are erased on a daily basis.

V. Grading Guidelines:

- This lab is worth 10 points. Your lab grade is assigned based on completeness and effort. To receive full credit for the lab you must show up on time and continue to work on the problems until the TA has dismissed you.