CMPS 1600 Introduction to Computer Science II – Fall 2018

Lab 6

This lab is due on Monday, November 5th. The goal of the lab is to practice the basics of C.

Objectives

- To practice basics of C programming and creating linked structures in C.

If you haven't gotten a chance to install C on your computer, please check the course website for possible online environments.

Guessing game

Input validation

To warm-up with C, write a program with a function `int ask_in_range(min, max)` that asks the user to enter an integer number in the range from `min` to `max`, inclusive. If the entered number is within the valid range, the function should return it; if not, it should keep asking until the user enters a valid number. To get started, see sample C programs in Zybook and slides on Canvas. A sample run of your program may look like follows:

```
Please enter a number: 1999
Your number is outside of [-100, 100] range.  Please enter a number:  -2
Your number is outside of [-100, 100] range.  Please enter a number:  99
```

Guessing game

Write a function `int guessing_game(num, rangemin, rangemax)` that takes an integer and plays a guessing game with the user. Use the `int ask_in_range(min, max)` to get the user to guess a number `num`. If the guess is too low or too high, let the user know and ask for another guess. Keep asking for guesses until the user enters a correct guess that equals `num`. Keep track of number of guesses that the user took, and output that number when the game ends. Also make your `guessing_game(num, rangemin, rangemax)` function return that number.
An example run may look as follows:

```
Hello and welcome to the game.
You need to guess a number between -100 and 100.
Please guess a number: 150
Your number is outside of [-100, 100] range.  Please enter a number:  50
Too high!
Please enter a number: 25
Too high!
Please enter a number: 10
```

```
Too low!
Please enter a number: 15
Too high!
Please enter a number: 13
Good job! You took 5 guesses.
```

Guessing game with randomness and history

Now you will augment your guessing game.

Adding randomness

Search the web and find how to generate a random integer number in a given range. Write a function `int get_random(rangemin, rangemax)` that takes range boundaries and returns the random integer within that range. Rewrite your guessing game so that it uses your `int get_random(rangemin, rangemax)` function to generate `num` parameter for the `guessing_game()`. Now enjoy playing the game you just wrote. What is the maximum number of guesses one will need to guess a number for the range [-500, 500]? Write the answer in a comment below your code.

Adding history

To make the guessing game even more fun, it would be nice to motivate the user play the game more by teasing them with a score (number of guesses) achieved by another player shown at the end of the game. To enable this, the game should store the previous players name and score in a file "history.txt". It also should ask the user for their name in the beginning of the game, and overwrite previous value with this user's name and score when the game is over. Add this functionality to your game.

Linked structures and pointers

Warmup

Write a small program that calls following two functions. Use the program to understand the diferences in their behaviors.

```
void swap(int x, int y)
{
int temp;
temp = x;
x = y;
y = temp;
return;
}
void swap(int \*x, int \*y)
{
int temp;
temp = \*x;
```

```
\*x = \*y;
\*y = temp;
return;
}
```

In your own words, explain how how the * and & operators work.

Linked List

1. Define a struct that stores a node in memory. You will want to use 'typedef' so your struct becomes a new type it will make your life easier. Think about what you need in the struct - some data, and then a reference to another node. What are the types?

2. In the main function, use malloc to allocate memory to hold a node. Set this nodeś data to hold the number 1. Do the same thing for another node that holds the number 2. Link these nodes together, 1 in front of 2. Using only the pointer to the first node, print out the data of both nodes.

3. Define a function to add a new node to the beginning of the linked list. Use malloc to create memory for the new node.

4. Define a function to remove the first node from a linked list. Make sure to use free to free the memory of the removed node.
   There are multiple ways of writing these functions. Two possible approaches are:

   - Write functions that take a pointer to a node (the head), and return a pointer to the new head of the list after you've modifed the structure.
   - Write functions that return void, but take a pointer to a pointer to a node: the **node object you get will allow you to modify the pointer itself within the function body though you will have to call the functions with the & operator applied to the pointer to your struct.

You are free to choose any approach you like. Make sure to explain your approach with comments.

Accessing members of a structure

A distance measure can be expressed by two parts one part is in inch and other in feet. Define a struct called "distance" that stores the information about each instance variable. Then write a C function called "addDistance" that receives two input parameters of type "distance" and adds them and returns a new parameter of type "distance". Write a main program in which the user inputs two distances and adds them using the addDistance function and prints the outcome on the screen.

```
Output:
1st distance
Enter feet: 12
```

```
Enter inch: 7.9
2nd distance
Enter feet: 2
Enter inch: 9.8
Sum of distances = 15'-5.7"
```

Submission

Submit 4 files - one with a basic guessing game, one for guessing game with randomness, and memory, one with code for linked structures, and one for Accessing members of a structure.