Rec14 - Recursion

May the Fourth be with you.

Topics

- recursion

Recitation Instructions

- Solve the tasks below *using recursion*. They should all be fairly straight forward.
- Once you have tested your code and are confident that it is working correctly, *then run each of your functions through the debugger* for your system and examine the call stack as you go.
  Usually we use the debugger to [duh] debug our programs. Here you are using it to understand the way your code works. The key to understanding recursion is to understand recursion... No, no. I meant to say the key is understanding how the call stack represents each step of the recursion.

Ok, here are the things to do today. Hopefully by the time you are done, you will be an ace at recursion.

1. Write a recursive function to determine if a non-negative integer has an even number of ones in its binary representation. Return true if it does and false otherwise.
   - Note that you are *not* being asked to compute the *number* of bits that are one, only if that *would* be even.
   - Think about what the recursive call will do for you. It will tell you if the subproblem had an even number of bits. Now you just need to take that information and combine it with whether this next bit you are looking at is one or not.
   - And **do not** pass any additional parameters. They are not necessary and they do not make the problem easier.
2. Write a recursive function to create and return a new list that is the sum of the values in the the two lists passed in. Do *not* assume that the two lists are the same length. (By *a list*, I mean a `Node*`, not an instance of a list class.)
3. Write a recursive function to return the maximum of the values in a binary tree.

- We will assume there is no meaningful value for the maximum of an empty tree. You should throw an appropriate exception for C++'s exception heirarchy (http://www.cplusplus.com/reference/exception/exception/).
- Note that the standard exceptions are in a header file called stdexcept. Also, they tend not to have a default constructor, but rather to *require* a string argument that will be returned by the method `what()`.
- Be sure to *test* your code using try / catch and demonstrating that your function works correctly whether the tree passed is empty or not.
- Note, this is **not** a binary search tree.

4. Write a recursive function called palindrome. It will be passed an *array* Note that when an array is passed to a function, only the address of the first element is passed. So, what your function is really being passed is the starting point of the array and how many characters there are. Of course, to use it you can just pass in the name of the array, as the name is equivalent to its address.
   Your recursive calls only need to be passed the address of the portion of the array you want to process and the number of characters.
   An example of calling your function might be:`char s[] = "racecar";`
5. `if (palindrome(s, 7)) { cout << "Yes!\n"; }`
6.

   It of course would also return true for "noon".
7. Implement a recursive function to solve the towers of hanoi problem. However, your function should **not print** anything. Instead it should just return the *number of moves* needed.Remember: 1) you are writing a recursive function. We are not interested in seeing a function that just outputs the result of computing a formula. 2) As always, **no globals**.
   If you are unsure what the results should be, the test code that I used to check my solution generated the following output:
   `#5) Testing towers`
8. `towers(1, 'a', 'b', 'c'): 1`
9. `towers(2, 'a', 'b', 'c'): 3`
10. `towers(3, 'a', 'b', 'c'): 7`
11. `towers(4, 'a', 'b', 'c'): 15`

12. `towers(5, 'a', 'b', 'c'): 31`
13. `towers(6, 'a', 'b', 'c'): 63`
14. `towers(7, 'a', 'b', 'c'): 127`
15. `towers(8, 'a', 'b', 'c'): 255`
16. `towers(9, 'a', 'b', 'c'): 511`
17. `towers(10, 'a', 'b', 'c'): 1023`
18.

19. Given the following function definition, *figure out* what it will return for values 0 to 10, and verify that you were correct. (If you were not correct, the use your debugger or print statements to determine where you went wrong.

```
void mystery(int n) {
   if (n > 1) {
      cout << 'a';
      mystery(n/2);
      cout << 'b';
      mystery(n/2);
   }
   cout << 'c';
}
```

Additional resources for assignment

- tree.cpp (1 KB; Jan 2, 2018 2:39 pm)