CSE 232 Fall 2017

Programming Project 04

This assignment is worth 30 points (3% of the course grade) and must be **completed and turned in before 11:59 on Monday, October 2nd .**

Assignment Overview
This assignment will give you more experience with functions and a little work with strings

Background
Basically, we are going to play games with an integer number and its sequence of digits. We can classify numbers as to the order of its digits and their relationship. There are 4, rather odd, categorizations which are:

- **metadrome**: An integer whose digits are **strictly** increasing: 12345 is a metadrome, 11223 is not.
- **plaindrome**: An integer whose digits are either the same or increasing: 11223 is a plaindrome, 11221 is not
- **katadrome**: An integer whose digits are **strictly** decreasing: 54321 is a katadrome, 54432 is not.
- **nialpdrome**: An integer whose digits are the same or decreasing: 554432 is a nialpdrome, 54323 is not.
- **nondrome**: If an integer is not classified as any of the above, it is a nondrome.

Different Bases to a number
We are also going to allow you to make the above distinctions in the context of a numeric base. For example, if the number were base 2, then only 0,1 are allowed in the integer. If the number were base 16, then 0-9 and a-f would be allowed. We are going to extend that idea all the way to base 36 by allowing the lower case letters a-z to signify a multidigit number: a == 10, b ==1, … f == 15, …, z == 35.

Project Description / Specification
Reminder
Just a reminder. We provide *exactly* our function specifications: the function name, its return type, its arguments and each argument's type. Do not change the function declarations!

Functions

function: `metadrome`: Boolean return. Arguments are

- a `string n`, the input number, and an `int base`.

If `n` is a metadrome in the provided `base` the function returns `true`, otherwise it returns `false`.

function: `plaindrome`: Boolean return. Arguments are

- a `string n`, the input number, and an `int base`.

If `n` is a plaindrome in the provided `base` the function returns `true`, otherwise it returns `false`.

function: `katadrome`: Boolean return. Arguments are

- a `string n`, the input number, and an `int base`.

If `n` is a katadrome in the provided `base` the function returns `true`, otherwise it returns `false`.

function: `nialpdrome`: Boolean return. Arguments are

- a `string n,` the input number, and an `int base`.

If `n` is a nialpdrome in the provided `base` the function returns `true`, otherwise it returns `false`.

function: classify: string return. Arguments are

- a `string n,` the input number, and an `int base`.

returns one of 5 strings: metadrome, palindrome, katadrome, nialpdrome, nondrome. It returns the most specific classification it can. For example, 12345 is both a metadrome and a palindrome. Since metadrome is more restrictive, that is the string returned. If no classification applies, the function returns nondrome.

main : ***<u>Very Important!</u>*** We provide the main and a global constant `base_vals` (see below in Hints) in `proj04/proj04.cpp`. These files will be the starter code. When you submit, the main should be exactly as in the starter code. Do whatever you need to main for testing before submission, but the main that gets turned in needs to be the main provided in the starter code. If you modify the main during submission you will receive a 0 for the project

Input and Output
We provide 6 test cases in Mimir for you to test. 2 other points will be assigned by the TA upon inspection of the code.

Deliverables
In Mimir under `Project 04: the dromes` you will find `proj04/proj04.cpp` This time `proj04.cpp` is not empty! The when you submit to mimir you should use this main.

General Hints:

1. You can write as many functions are you like over and above the ones I have specified.
    a. Make sure you write the requested functions exactly as specified. They will be tested individually according to that specification.
2. All but one of the functions return Booleans, which is not very informative. Feel free to place lots of output statements in your functions so you can see what is going on.
    a. Just remember to remove the output statements before you turn in the code!!!
3. The main will test each function. You already have the main, so develop the functions one at a time and run the appropriate test case to see that the function works!
    a. Don't write everything all at once, write one function at a time, test it and make sure it works by running the appropriate case against it.

Specific Hints
Do what you think is right, but here are some hints to help.

1. Dealing with these input numbers as strings makes things a lot easier.
2. If you do use strings, you might discover that the string `find` function and string indexing will make your code much easier to write.
3. Solving for one "drome" makes getting the rest write easier. Work on one of them, get it right, then move on to the others.
4. How to deal with different bases? The global in the skeleton file `base_vals` is a string that can help with that. The problem is to restrict a particular base to the appropriate digits/letters. However, that is really just an issue of indexing into this string. For example, base 2 numbers should only use '0' or '1'. However, if one substrings into `base_vals` using `0,2` (that is, starting at index 0,

length 2), that substring contains the legal characters. Base 10, substring 0,10 gives 0-9. Base 16, substring 0,16 gives 0-9 and a-f. The max base we will allow is the length of `base_vals`.

5. If an "illegal" character, say the number 8 in a base 2 number occurs, then:
   a. all of the "dromes" will return `false`.
   b. classify should return a `nondrome`.