CSC 120 (R Section)— Lab Exercise 2

This is a non-credit excercise, which you do not hand in.
You may work on your own or together with another student, as you please.

In this lab, you will write several small R functions that manipulate vectors. You should put the definitions of these functions in an R script file called `lab2defs.R`. You can "source" this script to define the functions, and then test them by typing calls of these functions with various arguments. You should start with just the first function below, test it and modify it until it is working, then go on to writing the next function, and so on.

You should write all these functions using only the facilities of R that we have covered in lectures (through week 3), or that are suggested below, even though some of these problems could be solved more easily using facilities of R that we haven't covered yet.

1) Write a function `sum_first_last` that takes a numeric vector as its only argument and computes the sum of the first and last elements of this vector. For example, `sum_first_last(c(3,1,2,1,8))` should be `11`.

   For this exercise, you'll need to use the `length` function, which gives how many elements are in a vector.

2) Write a function `midvalue`, which takes a numeric vector as its only argument. If the length of the vector is odd, it returns the middle element of this vector. If the length of the vector is even, it returns the average of the two middle elements. For example, `midvalue(c(7,1,3,7,2))` should be `3`, and `midvalue(c(7,3,7,2))` should be `5`.

   For this exercise, you'll need to use the `%%` operator, which finds the remainder of a division. For example, `8%%3` is `2`, and `8%%2` is `0`.

3) Write a function `sum_positive` that takes a numeric vector as its only argument, and computes the sum of the positive (greater than zero) elements of this vector. For example, `sum_positive(c(7,-1,0,2))` should be `9`.

   Hints: Set a variable `s` to zero, and then do a `for` loop that looks at each element of the vector, adding that element to `s` if it is positive. Finally, write just `s` as the last step in the function, so that the final value of `s` will be the function's value.

4) Write a function `swap_first_last` that takes a vector (of either numbers or strings) as its only argument, and returns a modified version of this vector in which the first and last elements are swapped. For example:

```
> x <- c("fred","mary","george","bert","helen")
> swap_first_last(x)
[1] "helen"  "mary"   "george" "bert"   "fred"
```

5) Write a function `find_cos_integer` that takes a number, x, as its only argument and returns the smallest non-negative integer, i, such that $|\cos(i) - x| < 0.001$. Here is an example:

```
> find_cos_integer(0.1234)
[1] 38282
> cos(38282)
[1] 0.1224119
```

For this exercise, you will need to use the `abs` function, which gives the absolute value of its argument.

What happens when you try `find_cos_integer(1.2)`? You may find the red STOP sign in RStudio to be useful!

6) Write a function `count_char` that takes two arguments, a vector of strings, `str`, and a string consisting of a single character, `chr`, and returns a vector of numbers the same length as `str` that gives the number of times `chr` occurs in each of the strings in `str`. For example:

```
> count_char( c("fred","mary","george","bert","helen"), "e")
[1] 1 0 2 1 2
```

For this exercise, you will need to use the `nchar` function, which gives the number of characters in a string, and the `substring` function mentioned in the week 1 lecture slides. You will also need to use the `numeric` function, which creates a numeric vector of a given length (containing all zeros).