CS 31 Worksheet 2

This worksheet is entirely **optional**, and meant for extra practice. Some problems will be more challenging than others and are designed to have you apply your knowledge beyond the examples presented in lecture, discussion or projects. All exams will be done on paper, so it is in your best interest to practice these problems by hand and not rely on a compiler.

Concepts

While Loops, Do While Loops, Functions -- by value and by reference, Switch Statements

Reading Problems

1) What does the following code snippet output?

```
void mystery(int& a, int b) {
      int count = 0;
      while (count < 2) {
            a = a + b/2;
            b = a + 5;
            cout << "a: " << a << " b: " << b << endl;
            count++;
      }
}

int main() {
      int a = 5, b = 10;
      cout << "a: " << a << " b: " << b << endl;
      mystery(a, b);
      cout << "a: " << a << " b: " << b << endl;
}
```

2) What does the following code snippet output?

```
void mystery(char code) {
  switch(code) {
    case 'a':
    case 'b':
    case 'c':
      cout << "spooky";
```

```
      break;
    case 'd':
      cout << "feeling";
      break;
    case '1':
      cout << " ";
      break;
    case '2':
      cout << "?";
    default:
      cout << endl;
      break;
  }
}

int main() {
  string message = "d1a2c1d#";
  int i = 0;
  do {
    mystery(message[i]);
    i++;
  } while(i != message.size());
}
```

Programming Problems

1) Create a function *changeString* that accepts two parameters:
   - string1: a reference to a string value that does not contain spaces and
   - string2: a string consisting of letters that will be used as delimiters.

   Now, for every character in string2 that appears within string1, replace the letter within string1 with a space.

   Note: You may assume that every letter within string2 will be unique.

   For example:
   changeString("Helatelmylcookie", "l") -> "He ate my cookie"
   changeString("ShouldeHIstartemylab?", "He") -> "Should I start mylab?"

2) a) Write a function *isPalindrome* that takes in a string and determines if it is a palindrome. A palindrome is a string that reads the same forwards and backwards.

For example:
isPalindrome("abcba") returns true
isPalindrome("skt_will_win") returns false
isPalindrome("z") returns true
isPalindrome("") returns true

b). Now write a function, *isPalindrome2,* that is similar to the function above, except we don't care about spaces in the string.

For example:
isPalindrome2("ggnore ero n g g") returns true

3) Write a function *findRun* that takes in a string of lowercase and uppercase alphabetical characters and returns the character with the longest "run." In other words, return the character that occurs the most times in succession. You may assume that the string is not empty. If two characters have equally long runs, return the first one.

For example:
findRun("abbccccdda") returns 'c'
findRun("aaaabcbbbbcbcbcbcbcb") returns 'a'

4) Write a function *integerDivide* that does integer division without using the division operator (/). You can assume both parameters will be positive.

For example:
integerDivide(6, 2) returns 3

5) Write a function *findLastLength* that takes in a string that consists of uppercase alphabetical characters, lowercase alphabetical characters, and empty space ' ' characters. It returns the length of the last word, unless the last word does not exist, in which case it returns 0.

For example:
findLastLength("Misfits should have won against SKT") returns 3
findLastLength(" ") returns 0

6) Write a function *intPalindrome* that returns whether or not two integers are palindromes. They must be exact palindromes, not 10 and 01 (because 01 == 1), etc.

For example:
intPalindrome(62, 26) returns true

intPalindrome(154, 451) returns true
intPalindrome(25, 56) returns false

7) Write a function that takes a string representing an english sentence as a parameter and returns a string representing that sentence translated into pig latin. Here are the rules for input and translation:

Input: A string representation of a sentence. The sentence contains words separated by spaces, some of which may be have a capitalized first letter. The sentence ends with a period directly after the last word. You may assume the sentence contains only letters, spaces, and one period at the end.

Ex: "David Smallberg is my favorite professor."

Translation: A sentence can be translated into pig latin word by word, following these rules:

1. If a word in english starts with a vowel, its pig latin translation is simply the english word with "ay" added at the end.
   Ex: "apple" => "appleay"
2. If a word in english starts with a consonant, that consonant is moved to the end of the word and then "ay" is added at the end.
   Ex: "chapter" => "haptercay"
3. If a word has a capitalized first letter, its pig latin translation should also have a capitalized first letter.
   Ex: "David" => "Avidday"
   Ex: "Eggert" => "Eggertay"

Here is an example translation. Feel free to write any helper functions you may find useful in implementing your pig latin translator function.

Ex: "David Smallberg is my favorite professor." =>
"Avidday Mallbergsay isay ymay avoritefay rofessorpay."