Nathan Tung
Com Sci 31 (Lec 2)
October 23, 2011

CS31 Project 3 Report

a. Some notable challenges I faced while during this project include figuring out how to run the main function while checking whether to classify another email (and hence run it again). To solve this, I used a do-while loop that would cause the main function to check at least one email; at the end, if the user's input satisfies the condition (where it matches the string "y"), the loop runs again for the next email.

   One of the first challenges I faced related to coding the functions for returning words from a string. It was difficult at first to understand how to extract words (that is, alpha characters surrounded by symbols or the string boundaries) from a string. However, I realized that by coding and using getFirstWord in a loop, I could complete extractWord. In general, I learned that it is much easier to break down a somewhat complicated function into smaller, simpler bits of code and code them separately.

   Lastly, I thought accepting the user's input for the body e-mail over several lines was somewhat tricky. My solution was to accept a new temporary string for each line and concatenate it to a main string (with a space in between) until the user inputted a blank string – that is, he/she pressed enter twice in a row.

b. The main function initiates the counter for spam messages and legitimate messages. Then it delves into a do-while loop in which a spam score is initiated to zero, and the user is asked to input a subject string and a body string. Then the inputted email is passed through a series of five spam tests. The first test compares the ratio of all-capitalized words in the subject to all words in the subject. The second test checks if the last word in the subject ends with multiple (three) consecutive consonants. The third test checks if there are consecutive (three) exclamations. The fourth test repeats the first test, but for the body of the email, and with a lower threshold ratio (.5 as opposed to .9) for considering the email as spam. The fifth and last test totals together all words that are considered "special" and indicate spam. Each of the five tests add to the spam score accordingly.

   If the score exceeds 100, then that email is deemed spam. Otherwise, it is legitimate. The user must then input "y" to rerun the do-while loop for another email or input "n" to see the summarized tally of spam emails to legitimate emails. Any other input by the user will cause the program to prompt for the only acceptable input.

   All functions (briefly summarized below) except for the test case are used in the main function.

Nathan Tung
Com Sci 31 (Lec 2)
October 23, 2011

main():
- Initiate spam and legitimate email counter
- Do-While loop:
  - Initiate spam score to 0
  - Prompt user for email subject; store into string
  - Prompt user for email body; concatenate lines into string
  - Do first test for spam:
    - While there is a word in subject, extract it:
      - Compare that word to its all-uppercase version
    - See if ratio of uppercase words to all words is over 0.9
      - If so, add 30 points to spam score
  - Do second test for spam:
    - Get last word in subject and check if it is gibberish
      - If so, add 40 points to spam score
  - Do third test for spam:
    - Check if there are multiple exclamations (3 in a row) in subject
      - If so, add 20 points to spam score
  - Do fourth test for spam:
    - While there is a word in body, extract it:
      - Compare that word to its all-uppercase version
    - See if ratio of uppercase words to all words is over 0.5
      - If so, add 40 points to spam score
  - Do fifth test for spam:
    - While there is a word in body, extract it:
      - Add 5 points to spam score for each special word found
  - If spam score exceeds 100, deem that email spam; else, it's legitimate
  - Prompt user for repeating the process:
    - If "y", satisfies do-while condition and repeats
    - If "n", quits program after displaying spam and legit email counts
    - If anything else, keep prompting the user for "y" or "n"

getFirstWord():
- Initiate beginning index position for first word
- While index is in bounds and character there is not a letter:
  - Increase index position
- If no such letters are found, return a blank string
- While consecutive characters are letters
  - Count the positions
- Return substring starting at the initially-found index going for counted positions

Nathan Tung
Com Sci 31 (Lec 2)
October 23, 2011

getLastWord():
    Initiate ending index position for last word
    While index is in bounds and character there is not a letter:
        Decrease index position
    If no such letters are found, return a blank string
    While consecutive characters are letters
        Count the positions
    Return substring starting at the index going for counted positions until first index
extractWord():
    Find first word and store temporarily
    If there is no first word, return blank string
    Else, set string reference to substring from end of first word; return first word
isUppercase():
    For each character in given string:
        If any character is alpha and lowercase, return false
        Otherwise, return true
makeUppercase():
    Initiate a blank string
    For each character in given string:
        If any character is alpha, concatenate its uppercase version to the string
        Else, concatenate it to the string as is
    Return the string
hasMultipleExclamations():
    Initiate a total exclamation count and current count
    For each character in given string:
        If it is an exclamation point, add to current count
        Else, update exclamation count with current count and reset current
    Update exclamation count one last time with current count if necessary
    Return whether consecutive exclamation count is at least 3
isGibberishWord():
    Initiate a total consonant count and current count
    For each character in given string:
        If it's an alpha and non-vowel, add to current count
        Else, update consonant count with current count and reset current
    Update consonant count one last time with current count if necessary
    Return whether consecutive consonant count is greater than 3
isVowel():
    Return whether character is any capitalization of 'a', 'e', 'i', 'o', or 'u'
isSpecialWord():
    Return whether the string is a special word specified in the specs

Nathan Tung
Com Sci 31 (Lec 2)
October 23, 2011

c. The code in the function testAll() tests each of the required functions individually.

```
assert(isVowel('a'));
assert(isVowel('e'));
assert(isVowel('i'));
assert(isVowel('o'));
assert(isVowel('u'));
assert(isVowel('A'));
assert(isVowel('E'));
assert(isVowel('I'));
assert(isVowel('O'));
assert(isVowel('U'));
assert(!isVowel('B'));
assert(!isVowel('z'));
assert(!isVowel('!'));
assert(!isVowel('5'));
```

→Tests if all the vowels of all capitalization are covered by the function. Also tests other random characters, digits, and symbols to make sure they return false.

```
assert(makeUppercase("hello")=="HELLO");
assert(makeUppercase("HellO")=="HELLO");
assert(makeUppercase("hElLo")=="HELLO");
assert(makeUppercase("z814F")=="Z814F");
assert(makeUppercase("0!!!")=="0!!!");
assert(makeUppercase("bigger")=="BIGGER");
assert(makeUppercase("na~h")=="NA~H");
assert(makeUppercase("")=="");
```

→Tests to make sure all alpha characters translate into capitalized letters. Symbols, digits, and strings without alphas are included to make sure nothing throws off the code.

```
assert(isUppercase("WOW!!") );
assert(isUppercase("NOOOOOOOOOO!!$%!^@&*#") );
assert(isUppercase("W0WZERZ~~#084") );
assert(!isUppercase("NOOOOOOOOOooOOOOo") );
assert(!isUppercase("B000000`13`!~o") );
assert(!isUppercase("WoW!!") );
assert(isUppercase("") );
assert(isUppercase("99") );
assert(isUppercase("!!") );
```

→Tests to make sure ALL the characters are capitalized, and that digits or symbols do not affect the code.

```
assert(hasMultipleExclamations("!!!"));
assert(hasMultipleExclamations("afh!!akjg!!!jgu83tbi!!!!!!8bga"));
assert(hasMultipleExclamations("something!!hereee!!!!!ee"));
assert(hasMultipleExclamations("string text!!!!!!"));
assert(!hasMultipleExclamations("!!"));
assert(!hasMultipleExclamations("!!-!!-!!b8"));
assert(!hasMultipleExclamations("!!1!946153!!487!~!!*!!"));
assert(!hasMultipleExclamations("nothing here"));
assert(!hasMultipleExclamations(""));
```

→Checks to see if the exclamation counter is working properly, and that strings without any exclamations do not return true.

```
	assert(isGibberishWord("aaaaabnfjanuba"));
	assert(isGibberishWord("ngaonrgoangsfas"));
	assert(isGibberishWord("11876889~ffdf8*78~"));
	assert(isGibberishWord("fghj"));
	assert(!isGibberishWord("jkl"));
	assert(!isGibberishWord("ajkl~"));
	assert(!isGibberishWord("ccac"));
	assert(!isGibberishWord("sadde"));
	assert(!isGibberishWord("`123456789875432@#$%"));
	assert(!isGibberishWord("cool dude"));
	assert(!isGibberishWord("NASA"));
	assert(!isGibberishWord(""));
```

→Checks to see if the consonant counter is working properly, and that strings without any consonants do not return true.

```
	assert(getFirstWord("hello there") == "hello");
	assert(getFirstWord("what is there to talk about") == "what");
	assert(getFirstWord("947`7~&*&$ ~(*$~") == "");
	assert(getFirstWord("") == "");
	assert(getFirstWord("              ") == "");
	assert(getFirstWord(" !!! hello !!!") == "hello");
	assert(getFirstWord("`123456789findme lol") == "findme");
	assert(getFirstWord("???pro???blem???") == "pro");
	assert(getFirstWord("`12345689`765432185@#$%^*(9 I wonder if this will
work") == "I");
	assert(getFirstWord("1234NoThing is wrong with this!") == "NoThing");
```

→Tests to see if words can be extracted amidst symbols and digits, and whether blank strings or digits and symbols alone influence the program. Tests for the first words.

```
	assert(getLastWord("hello there") == "there");
	assert(getLastWord("what is there to talk about") == "about");
	assert(getLastWord("947`7~&*&$ ~(*$~") == "");
	assert(getLastWord("") == "");
	assert(getLastWord("              ") == "");
	assert(getLastWord(" !!! hello !!!") == "hello");
	assert(getLastWord("`123456789findme lol") == "lol");
	assert(getLastWord("???pro???blem???") == "blem");
	assert(getLastWord("`12345689`765432185@#$%^*(9 I wonder if this will
work") == "work");
	assert(getLastWord("1234NoThing is wrong with this!") == "this");
```

→Tests to see if words can be extracted amidst symbols and digits, and whether blank strings or digits and symbols alone influence the program. Tests for the last words.

```
	string s = "***hello there";
		assert(extractWord(s) == "hello"  &&  s == " there");
		assert(extractWord(s) == "there"  &&  s == "");
		assert(extractWord(s) == ""  &&  s == "");
	string x="`123456789findme lol";
		assert(extractWord(x) == "findme"  &&  x == " lol");
		assert(extractWord(x) == "lol" && x == "");
	string y=" !!! hello !!!";
		assert(extractWord(y) == "hello"  &&  y == " !!!" );
		assert(extractWord(y) == "" && y == " !!!");
```

```
string z="???pro???blem???";
       assert(extractWord(z) == "pro"  &&  z == "???blem???");
string j="          ";
       assert(extractWord(j) == ""  &&  j == "          " );
string k="";
       assert(extractWord(k) == ""  &&  k == "" );
```

→Verifies that, after finding the first word, it is removed from the string correctly without leaving behind traces of symbols or digits prior to it. Makes sure that a blank string does not produce an error. Tests show that the function works after several runs.

The program handles all of the test cases correctly. However, during development, getFirstWord, getLastWord, and extractWord were basically the only functions giving noticeable yet small errors. In order to keep them from giving an error, I placed a check to see whether the given string was empty. If the given string is empty, then I will return an empty string in all cases. I also had to fix extractWord so that the inputted string would be altered to include all spaces and symbols following the extracted word.

Testing the main function:

Subject: "Hello there!"
Body: "This is a legitimate email that should have a spam score of zero."
→There are no all-caps letters, consecutive exclamations or consonants, or otherwise special words that would increase the spam score.

Subject: "THERE ARE CAPITALS HERE"
Body: ""
→30 points should be added for capital words in the subject.

Subject: ""
Body: "THERE ARE CAPITALS HERE"
→40 points should be added for capital words in the body.

Subject: "Sex pill enlargement"
Body: "**Only** words here should be counted.
Free offer click now!"
→25 points should be added: 5 for each of the underlined words in the body only.

Subject: "asdf"
Body: ""
→The subject contains gibberish, thus 40 points should be added.

Subject: ""
Body: ""
→There is nothing inputted to warrant any more than 0 points.

Nathan Tung
Com Sci 31 (Lec 2)
October 23, 2011

Subject: "!!!"
Body: ""
→There are consecutive exclamations in the subject, so the score should be 20 points.

Subject: "HELLO THERE!!! QQQQ"
Body: "CONGRATULATIONS! You are a winner of the local lottery! **Click** here **now** for a **limited** time **offer**!"
→This should have a score of 110: 30 for all capital letters in the subject, 40 for gibberish, 20 for exclamation points, 0 for capital letters in the body, and 20 for special words.

Subject: "ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT NINE ten"
Body: ""
→This should give 0 points; the ratio of capitalized words in the subject is not greater than 0.9

Subject: "ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT NINE TEN eleven"
Body: ""
→This should give 30 points; the ratio of capitalized words in the subject is greater than 0.9

Subject: ""
Body: "ONE TWO THREE FOUR FIVE six seven eight nine ten"
→This should give 0 points; the ratio of capitalized words in the body is not greater than 0.5

Subject: ""
Body: "ONE TWO THREE FOUR FIVE SIX seven eight nine ten"
→This should give 40 points; the ratio of capitalized words in the body is greater than 0.5.

All main function tests were handled correctly as well.