ENGR 121: Computation Lab I

Programming Assignment 2

This assignment comprises of four problems. Submit functions for each of the problems by November 30, 2016, 11:59 pm. Detailed submission instructions are available towards the end of this document.

You must work on this assignment on your own. Submitted solutions must be your original work. Solutions copied from other students or from online sources will result in a grade of zero for the entire assignment.

1. (**10 points**) The sine function can be evaluated via the following infinite series:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

Write a function `approxSine` that takes as input two parameters, `x` and `threshold`, to perform the following task. Starting with the initial approximation $\sin(x) = x$, add terms one at a time to improve this estimate until

$$\left| \frac{\text{true value} - \text{approximation}}{\text{true value}} \right| < \text{threshold}.$$

Assume that the true value of sine is the one returned by the built-in **sin** function in MATLAB.

Your function must return two output values: the approximate value of $\sin(x)$ and the number of terms that were needed to obtain this value subject to the desired error threshold. The following is an example of function behavior for $x = \pi/5$.

```
>> x = pi/5;
>> true_value = sin(x)
real =
0.5878
>> threshold = 0.001;
>> [approx terms] = approxSine(x, threshold)
approx =
0.5878
terms =
2
>> threshold = 0.00001;
>> [approx terms] = approxSine(x, threshold)
approx =
0.5878
terms =
3
```

Note that the `terms` value returned by the function only counts the number of terms added to the starting approximation of $\sin(x)$.

2. (**10 points**) Write a function called `analyzeText` that accepts a string as input and (1) counts the number of words; (2) identifies the length of the longest word; and (3) identifies the greatest number of vowels (a, e, i, o, u) in a word. The function will be called as follows:

```
[numWords, maxWordLength, maxNumVowels] = analyzeText(text)
```

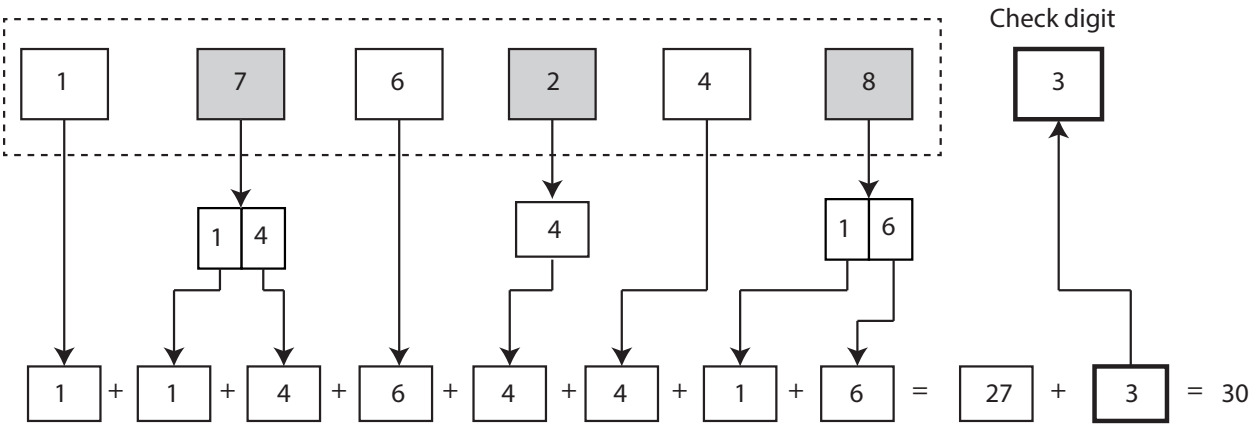
where the variable `text` contains the string supplied to the function, `numWords` returns the number of words, `maxWordLength` returns the length of the longest word in the string, and `maxNumVowels` returns the greatest number of vowels present in any word.

Following are examples of correct function behavior:

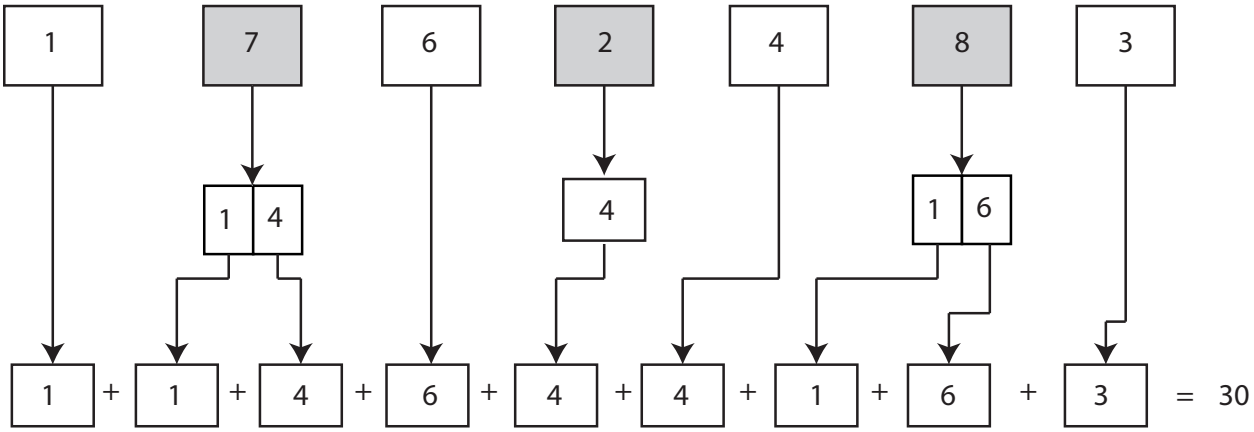
```
>> [nw, mwl, mnv] = analyzeText('The woods are lovely dark and deep')
nw =
7
mwl =
6
mnv =
2
>> [nw, mwl, mnv] = analyzeText('But I have promises to keep')
nw =
6
mwl =
8
mnv =
3
```

If you are using the **strtok** function to tokenize the string, read the release notes regarding its behavior: `https://www.mathworks.com/help/matlab/ref/strtok.html`

3. **(10 points)** You are asked to implement the following checksum formula to validate an identification number given to you. The formula works as follows. Using the original number, double the value of every other digit. Then add the values of the individual digits together (if a doubled value now has two digits, add the digits individually). The identification number is valid if the resulting sum is divisible by 10.



(a) Computing the check digit.



(b) Checksum validation.

Figure 1: The process of creating and validating the checksums.

Let's now walk through both ends of the checksum creation and validation process: computing a check digit and validating the result. Figure 1(a) illustrates the steps involved in computing the check digit. The original number, 176248, is shown in the dashed-line box. Every other digit, starting from the rightmost digit of the original number—which after the addition of the check digit, will be the second rightmost—is doubled. Then each digit is added together. Note that when doubling a digit results in a two-digit number, each of these digits is considered separately. For example, when 7 is doubled to produce 14, it is not 14 that is added to the checksum, but 1 and 4 separately. In our example, the generated checksum is 27; so the check digit is 3 because that's the digit value that makes the overall sum equal to 30. Recall that the checksum of the final number should be divisible by 10; in other words, the checksum should end in 0.

Figure 1(b) illustrates the process of validating the number 1762483, which now includes the check

digit. As before, we double every second digit starting with the digit immediately to the left of the check digit, and add the values of all digits, including the check digit, to determine the checksum. Since the checksum is divisible by 10, this number validates.

Write a function called `validateID` that takes as input an identification number and returns 1 if the number is a valid checksum, 0 otherwise; it also returns the checksum that is calculated. The identification number is supplied to the function as a string.

The function will be called as follows:

```
>> [valid, checksum] = validateID('1762483')
valid =
1
checksum =
30
>> [valid, checksum] = validateID('79927398713')
valid =
1
checksum =
70
>> [valid, checksum] = validateID('9347870082856823')
check =
     1
checksum =
    80
```

Note that you can use the **str2num** or **str2double** functions in MATLAB to convert each character in the input string to a numerical value. For example, one can convert the character '7' to the corresponding numeric value as `str2double('7')`.

4. **(10 points)** The *birthday problem* is as follows: given a group of n people in a room, what is the probability that two or more of them have the same birthday? It is possible to determine the answer to this question via simulation. Write a function called `calcBirthdayProbability` that takes as input n and calculates the probability that two or more of the n people will have the same birthday. To do this, the function should create an array of size n and generate n birthdays in the range 1 to 365 randomly. It should then check to see if any of the n birthdays are identical. The function should perform this experiment 10^6 times and calculate the fraction of time during which two or more people had the same birthday.

The function is called as follows:

```
probability = calcBirthdayProbability(numPeople)
```

Examples of correct function behavior follow:

```
>> numPeople = 10;
>> prob = calcBirthdayProbability(numPeople)
prob =
0.1175
>> numPeople = 20;
>> prob = calcBirthdayProbability(numPeople)
prob =
0.4117
>> prob = calcBirthdayProbability(numPeople)
prob =
0.4114
>> prob = calcBirthdayProbability(numPeople)
prob =
0.4113
```

Note that different runs of the program with the same value for `numPeople` yield slightly different probability results. This is expected since we are performing experiments with randomly generated values during each run of the program. The Cody tests will be designed to account for this type of variation in the results.

Submission Instructions

- Submit your solutions individually using the appropriate links on BBLearn under 'Assignments/Assignment 2' by November 30, 2016.
- In addition, you must submit and test all your answers on Cody by November 30, 2016. The submission site will be up and running by the middle of next week. We will send out an email to the class once the site is up.
- Test the code thoroughly on your local MATLAB installation before submitting to Cody for testing. In particular, if you are using `while` loops in your code, make sure there are no infinite loops. Note that Cody times out for infinite loops and in situations where the total processing time takes more than 60 seconds.