CS1133, Spr 2018, HW 3 (Lect 6, Pts=30+20)

Due February 13, 2018 (before 11:00 pm)

Problem 1 [30 pts]

The Universal Product Code (UPC) is a one-dimensional barcode symbology that is in wide use in many countries including the United States to identify a product at the point of sale.

The UPS-A consists of 12 numerical digits. The digits are represented by corresponding barcodes that are typically read by optical scanners. Here we only focus on the numerical digits. Here are examples of UPC-A barcodes of three products.

(a) Samsung Galaxy S9 Plus LifeProof Clear/Grey

(b) Samsung Galaxy S9 Plus LifeProof Clear/Black

(c) Campbell Creamy Tomato Soup, 15.5 oz., 12/pack

Figure 1: Examples of UPC-A Barcode Labels

Since scanners do not work properly all the time (notice how often a cashier has to scan an item more than once), safety measures are built-in to protect the accuracy of the information being read. The first digit determines the type of product. The next five digits represent the manufacturers unique identification number. The next 5 digits are used to identify the particular product, based on that manufacturers database. The final digit is the check digit.

The purpose of appending the check digit at the end is to facilitate determination of whether there were errors in reading the barcode. Its value is computed based on all the preceding 11 digits according to a special formula.

Suppose the 12 digits are stored in a row vector $\mathbf{d}$, then the check digit, d_{12}, is determined by the following equation.

$$3d_1 + d_2 + 3d_3 + d_4 + 3d_5 + d_6 + 3d_7 + d_8 + 3d_9 + d_{10} + 3d_{11} + d_{12} \equiv 0 \pmod{10}. \quad (1)$$

This means that the left-hand side must be an integer that ends with 0 (and so divisible by 10). This equation, involving modular arithmetic, must be obeyed by the digits of any UPC-A code.

This equation can be written as

$$d_{12} + s = 0 \pmod{10},$$

where

$$s = 3(d_1 + d_3 + d_5 + d_7 + d_9 + d_{11}) + d_2 + d_4 + d_6 + d_8 + d_{10}.$$

This equation involves modulo arithmetic with modulus 10. It can be solved to find the check digit x_{12}:

$$x_{12} = 10\,\lceil s/10 \rceil - s,$$

where $\lceil x \rceil$ is the ceiling function (round up to the next higher integer). In Matlab the function is named `ceil`.

It turns out that this scheme can always detect the presence of any single mistake in the UPC code. When such an error is detected, the scanner beeps and another scan has to be taken.

Here are 3 UPC-A codes obtained from the internet

```
6 6 0 5 4 3 4 4 6 3 9
6 6 0 5 4 3 4 4 6 3 3
0 5 1 0 0 0 1 6 4 6 1
```

written here without their check digits. To facilitate checking the correctness of your program, use the following statement to generate a certain number, m of additional UPC codes (also without their check digits):

```
nRandom = 6;
UPCR = floor(10 * rand(nRandom,nKnownDigits));
```

Here $m = 6$ and the number of known digits for the variable `nKnownDigits` can be obtained from the array of UPC codes given above. Then concatenate all the UPC codes to form an array of UPC codes to work with here.

In this problem, you need to find their check digits according to the method described above. To help you check and debug your program, we reveal to you that the check digits of the 3 UPC codes obtained from the internet are actually given by 2, 0, and 2, respectively.

You may find it better to first write your program to work with a single UPC code at a time. (There is no need to submit that part of your program.) When your program gives the correct check digits for each of the 3 cases with known check digits, then rewrite it so that it works with an array of UPC-A codes to find a vector containing their check digits. Some scalars will become vectors, and some vectors become 2D arrays. Certain operations, such as sums, may need special handling.

Display the check digits of the three UPC code obtained from the internet, as well as the corresponding ones computed by yourself. For the 3 UPC-A codes from the internet, you should produce the following results

```
The computed known check digits are:
     2
     0
     2

Known correct check digits from the internet:
     2
     0
     2
```

Finally, using Eq. (1), compute its results using all 12 digits of all the UPC codes. You need to perform the sums and take their modulus with 10. Display the result that you get. If everything is correct, you should get a 0 for all of them. This shows that the check digits for the UPC codes are indeed correct.

Similar schemes are routinely used to safeguard digital information such as credit card numbers, ISBN numbers of books, postal codes, etc. The most glaring exception is Social Security numbers. Any arbitrary nine numbers can be a legitimate Social Security number of a person in this country.

Problem 2 [20 pts]

From the symmetry of the above defining equation, Eq. (1), it is clear that instead of solving for the check digit given the remaining digits, it can be used in a similar way to compute any of the digits at even position if the remaining digits (including the check digit) are known. Therefore the above method for computing the check digit from the remaining digits can be used to compute any one of the other digits at even positions. This is useful when we know that one of the digits at even position is not scanned correctly. We can easily use the above method to compute its value based on all the remaining digits. We are not going to do that in this problem.

Instead, we focus on the digits at odd positions. Assuming here that m is a positive odd integer, we want to compute d_m from the other digits. It in clear that the unknown digit satisfies the equation

$$3d_m + s = 0 \pmod{10},$$

where s is 3 times the sum of all the other digits at odd position plus the sum of all the digits at even positions (including the check digit). This equation has only 10 possible solutions whose values depend on the values of s, as shown in the table here.

s	d_m
0	0
1	3
2	6
3	9
4	2
5	5
6	8
7	1
8	4
9	7

In this problem, work with all the UPC-A codes from the previous problem including the check digits that you have found. In case you are not able to compute the check digits, then use only the first 3 UPC codes, whose check digits were revealed to you. Have the user enter a positive odd integer, m, from 1 to 12. Imagine that the scanner was not able to reliably read correctly the digit at that position, but all the remaining once are actually correct, compute the digit at position m. Compare them with their actual values.

You should get similar output displays as shown in this example here:

```
Enter an odd digit between 1 and 12: >> 7
Computed missing digits:
     8
     8
     5
     1
     2
     8
     0

Actual missing digits:
     8
     8
     5
     1
     2
     8
     0
```