CSCI 1320 Computer Science I: Engineering Applications – Fall 2018
Instructor: Zagrodzki
Assignment 3
Due Sunday, September 23, by 6pm

Vectors and Matrices

Task 1 (6 points) Generate the following column vectors using

a) the colon operator
b) the linspace function.

[Note: Your submission should contain **two** expressions to generate each column vector, one using a) **and** another using b)]

```
colvec1 = -18
          -17
          -16
          -15
          -14
          -13
          -12

colvec2 = 8
          12
          16

colvec3 = 16
          10
          4
```

Task 2 (9 points) Find an ***efficient*** way to generate the following matrix (**do not** hardcode the values) **(6 points)**:

```
M =
     10   12   8
     13   9    4
     21   16   1
     29   3    2
```

Then, give expressions that will, for the matrix *M*,

A. Write the contents of the element in the last row, third column to variable *m1* **(1 points)**
B. Write the contents of the entire third column to vector *m2* **(1 points)**
C. Write the contents of the second and third rows to matrix M3 **(1 points)**

Task 3 (20 points) Consider the following vector *vec:*

```
vec = [-11 5 3 2 -18 4 -5 5 -66]
```

The vector *vec* erroneously stores negative values. We would like to eliminate those negative values so that we are left with the following values for *vec:*

```
vec = [5 3 2 4 5]
```

Your task is to determine where the negative values are and then to delete them from the elements of *vec*. You must present two methods to achieve the task:

a) using the built-in Matlab function **find** and create new vector called *vec1***,**
b) using a logical vector and create new vector called *vec2*.

Task 4 (15 points) Write a function that returns the transpose of a given 3x3 matrix **without using any of the built in functions** (e.g. transpose(M) or rot90(M), or the transpose operator ') **anywhere in your function***.* Call this function *transposemat,* it will receive on input argument x, which is a 3x3 matrix , and will return one output argument Y, which is a 3x3 matrix. Example:

```
>> x = [1 2 3; 1 2 1; 3 5 1];
>> y = transposemat( x );

>> transposemat([1 2 3; 1 2 1; 3 5 1])
ans =
3 by
     1     1     3
     2     2     5
     3     1     1


>> disp(transposemat([1 2 3; 1 2 1; 3 5 1]))

     1     1     3
     2     2     5
     3     1     1

>> help transposemat
transposemat(x) returns the transpose of matrix x
```

Note: The syntax for your function would be `transposemat([your_matrix])`

Task 5 (25 points) Easter Sunday is the first Sunday after the first full moon of spring. To compute the date, you can use this algorithm, invented by the mathematician Carl Friedrich Gauss in 1800:

1. Let y be the year (such as 1800 or 2001).
2. Divide y by 19 and call the remainder a. Ignore the quotient.
3. Divide y by 100 to get a quotient b and a remainder c.
4. Divide b by 4 to get a quotient d and a remainder e.
5. Divide 8 * b + 13 by 25 to get a quotient g. Ignore the remainder.
6. Divide 19 * a + b - d - g + 15 by 30 to get a remainder h. Ignore the quotient.
7. Divide c by 4 to get a quotient j and a remainder k.
8. Divide a + 11 * h by 319 to get a quotient m. Ignore the remainder.
9. Divide 2 * e + 2 * j - k - h + m + 32 by 7 to get a remainder r. Ignore the quotient.
10. Divide h - m + r + 90 by 25 to get a quotient n. Ignore the remainder.
11. Divide h - m + r + n + 19 by 32 to get a remainder p. Ignore the quotient.

Then Easter falls on day p of month n.

For example, if the value of y is 2001, then: a = 6, b = 20, c = 1, d = 5, e = 0, g = 6, h = 18, j = 0, k = 1, m = 0, r = 6, n = 4, and p = 15.

Therefore, in 2001, Easter Sunday fell on April 15.

Write a script *easterSunday.m* (without using any MATLAB toolboxes) that prompts the user for a year and prints out the year, month, and day of Easter Sunday. Here's an example of a possible sample run:

```
>> easterSunday
Please enter the year: 2001
In 2001, Easter Sunday fell on 4/15.
```

throwBall_func.m

Task 6 (25 points) In this week's lab, you were asked to create a script `throwBall.m`. Here, turn the code from a script into a function named `throwBall_func.m`. The function takes three arguments `v, theta` and `maxTime` as inputs and returns a logical value (`true` or `false`), depending on whether the ball hits the ground in the allowed time or not. Here is an example of a function call:

```
>> isTrue = throwBall_func(v, theta, maxTime)
```

While testing the script for Lab 3, you were asking the user to input values for `v` and `theta`. You probably noticed that for certain values of `v` and `theta` the ball

did not hit the ground in 20 seconds. Here you are evaluating whether the ball will hit the ground in `maxTime` seconds. If none of the values calculated for height (the vector `y`) are negative, your call to the function `find` will return an empty vector. Use this fact to figure out whether the ball hits the ground in the allowed time or not. Assign to the return argument variable the appropriate logical value (`true` or `false`).
Note: you must do this WITHOUT using selection statements (IF-ELSE).
Hint: you might want to investigate the function `isempty()`.

Submitting the assignment:
Make sure all your .m files are well commented and they include a comment block with your name, student ID, course number, assignment number and recitation section. Zip all the files together and submit the resulting *.zip* file through Moodle as Assignment 3 by due date.