ENGR 121: Computation Lab I

Sample Final Exam

1. Assume that we have the following vector that erroneously stores negative values. How do we eliminate those negative values? **Hint:** You may either use the concept of logical vectors or the `find` function.

```
vec = [11, -5, 33, 2, 8, -4, 25]
```

2. Find the sum of the first n terms of the harmonic series where $n > 1$ is an integer provided by the user:

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots + \frac{1}{n-1}$$

3. *Piecewise functions* are useful when the relationship between a dependent and an independent variable cannot be adequately represented by a single equation. For example, the velocity of a robot can be described by:

$$v(t) = \begin{cases} 10t^2 - 5t & 0 \leq t \leq 8 \\ 624 - 3t & 8 < t \leq 16 \\ 12e^{-0.1(t-16)} & t > 16 \\ 0 & \text{otherwise} \end{cases}$$

where the velocity v of the robot is a function of time t. Write a function `my_func` that takes as input a vector `t`, realizes the above piecewise function, and returns the corresponding output vector `v`. You may use MATLAB's built-in function **exp** to calculate the exponential.

4. The arithmetic mean μ is calculated by summing all of the elements of a given vector and dividing the answer by the number of elements. Generate a vector x of 10 random numbers picked from a uniform distribution. Use a `for` loop to calculate the sum of these numbers. Once you have the sum, divide it by the total number of elements in the vector to find the arithmetic mean. Do not use the built-in **sum** and **mean** functions in MATLAB. Now, the standard deviation σ is a measure of how spread out the data is from the mean value. The higher this value, the more the variation or spread in the data, and vice versa. Implement the following formula, also using a `for` loop, to obtain the standard deviation of the generated data:

$$\sigma = \sqrt{\frac{1}{N-1} * \sum_{i=1}^{N} \big(x(i) - \mu\big)^2}$$

Here μ denotes the arithmetic mean and N denotes the number of elements in your vector x. Once you have the answer, check for correctness: find the standard deviation of the vector using MATLAB's in-built function **std** and check to see if the answers match.

5. Write a MATLAB program that uses loops and selection statements as well as the output statements, `fprintf('#')` and `fprintf('\n')`, to produce a pattern of hash symbols shaped like:

```
#####
####
###
##
#
```

The above example is a 5×5 grid. Your code must work for any $N \times N$ grid.

6. If $|x| < 1$, it is known that

$$\frac{1}{1-x} \approx 1 + x + x^2 + x^3 + \cdots$$

Write a function `seriesapprox` to perform the following tasks. Starting with the simplest approximation $1/(1-x) = 1$, add terms one at a time to improve this estimate. After each term is added, compute the absolute error between the approximate and true values. (Use MATLAB to determine the true value.) Continue to add terms until this error is less than 0.0001. Your function must take x as the input and must return two output values: the approximate value of $1/(1-x)$ and the number of terms that were needed to obtain this value subject to the desired error.

7. The Taylor series approximation for $\log(1+x)$, where $\log$ is the natural logarithm and $|x| \leq 1$, is

$$\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \cdots$$

unless $x = -1$, since $\log(0)$ is undefined. Write a function `seriesapprox` that takes `x` and an error threshold value `epsilon` as inputs, and starting with the initial approximation of $\log(1+x) = x$, adds terms one at a time to improve this estimate until the error between the true and approximate values is less than the threshold. Use MATLAB's built-in natural logarithm function **log** to obtain the true value. Your function must return two output values: the approximate value of $\log(1+x)$ and the number of terms needed to obtain this value subject to the desired error. The function must also incorporate an error check in that if $x = -1$ or if $|x| > 1$, it must return $-\infty$ (or `-inf`) as the approximate value.

8. Develop a vectorized version of the following code:

```
function y = myfunc()
x(1) = 0;
y(1) = 2*x(1) + 1;
for i = 2:5
    x(i) = x(i-1) + 2;
    y(i) = 2*x(i) + 1;
end
end
```

The vectorized version of the code should replace the above **for** loop structure with MATLAB's built-in vector operators.

9. Write a function `mymedian` that will receive a vector as an input argument, and will sort the vector and return the median. Any built-in functions may be used, except the median function. Loops may not be used.