# CS 1132 Spring 2017 Assignment 1a due 2/15 at 11:59 pm

*Adhere to the Code of Academic Integrity.* You may discuss background issues and general strategies with others and seek help from course staff, but the implementations that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is never OK for you to see or hear another student's code and it is never OK to copy code from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

When submitting your assignment, follow the instructions summarized in Section 3 of this document.

| Do not use the `break` or `continue` statement in any homework or test in CS1132. |
|---|

## 1 Weekly Temperature Data in Ithaca

This assignment will reinforce your understanding of user-defined functions in Matlab, 1-d arrays, loops, and conditionals, through a simple example involving weekly temperature data. You will write code to generate artificial data and functions to analyze the data or extract meaningful trends. We will use the fact that a year has 52 weeks and that each of the 4 seasons comprises 13 weeks.

You will also complete the script `assignment1a.m` that is provided in order to test the functions you have written.

### 1.1 Generating Artificial Data

Implement the following function to generate artificial weekly temperature data for Ithaca:

```
function weeklyWeather = generateWeatherData(n_weeks,season1,season2,season3,season4)
% Generate a vector of length n_weeks of integer values, each representing
% the temperature (deg F) of one week in Ithaca.  n_weeks is the number of
% weeks from the start of the year.  season1,...,season4 are each a vector
% of length 2 specifying the temperature bounds of each season, where
% the first value in the vector is the lower bound in a season and the
% second value is the upper bound.
%
% The values in weeklyWeather are generated such that the first 13 values
% are random integers in the range [season1(1),season1(2)], the next 13
% values are random integers in the range [season2(1),season2(2)], ...,
% etc., up to n_weeks values.  Assume 2 <= n_weeks <= 52.
```

You may use built-in function `randi` to generate a random *integer*. Search the MATLAB documentation or type in the Command Window `help randi` or `doc randi` to learn how to use `randi`.[1]

After implementing the `generateWeatherData` function, add code to the `assignment1a.m` file to call the function. Does it work correctly? Check to make sure!

### 1.2 Investigating a Trend

We say that a weekly temperature data set is *perfect* if the weekly temperature values are strictly increasing for the first 2 seasons and strictly decreasing for the last 2 seasons. In other words, as we transition from winter to summer, we expect the weather to be warmer and as we transition from summer to winter, we

[1] Instead of `randi`, one could use `rand` to generate a random *real* value and then convert it to an integer. E.g., to get a random integer in the range $[a \ldots b]$ where $a$ and $b$ are integer values and $a < b$, one can use the expression `floor(rand*(b-a+1))+a`, i.e., `floor(rand*`*number_of_possibilites*`)+`*base_value*

expect the weather to be cooler. However, as can be observed from the weekly temperature values in Ithaca, this scenario rarely happens.

Implement the following function as specified.

```
function out = isPerfect(weeklyWeather)
% out is 1 if the first 26 values in vector weeklyWeather are strictly
% increasing and elements 27 to 52 are strictly decreasing; otherwise out
% is 0.  Vector weeklyWeather has a length >=2 and <= 52.
```

Please note that the number of weeks in the input `weeklyWeather` is not necessarily 52. If the number of weeks at most 26, it suffices to check that the temperature values are strictly increasing. Otherwise, you also need to check that the temperature values are strictly decreasing from week 27 onwards.

After writing the function `isPerfect`, call the function in `assignment1a.m` on the data set you generated from Section 1.1. In addition, test your `isPerfect` function on the weekly temperature data we have provided in `assignment1a.m`. You should test your function on other data sets of different lengths to check the accuracy of your code.

## 1.3 Finding Extreme Values

If the data set is very large, it is useful to extract meaningful subsets in order to facilitate the analysis of the data. Here we are interested in identifying extreme temperature values which can be useful in mitigating risks. Implement the following function as specified.

```
function [weekIndex,temp] = findExtremeTemp(weeklyWeather,thresh)
% Extract from vector weeklyWeather the temperature values above a
%   threshold.
% weeklyWeather: a non-empty vector with a length <= 52
% thresh: threshold at or above which to extract data
% temp: a vector, possibly empty, containing the extracted temperatures
% weekIndex: the vector, possibly empty, storing the indices of the
%            extracted temperatures corresponding to vector temp
```

You cannot use the following built-in functions in Matlab to write `findExtremeTemp`: `max`, `min`, `sort`, and `find`. Once you have written the function, again call the function in `assignment1a.m` on the data generated in Section 1.1 for your choice of `thresh` to test your work.

## 1.4 Cleaning the Data

In some situations, the data collected may have been corrupted by human error or problems with the measuring device, etc. In this example, we are concerned with the scenarios in which some of the temperature values are negative. This can happen if the person gathering the data recorded the temperature in terms of degrees Celsius, for example. Implement the following code as specified.

```
function weeklyWeather = cleanData(weeklyWeather)
% Interact with the user in order to change any negative values in
%   weeklyWeather into non-negative values.
% For each value in vector weeklyWeather that is negative, prompt the user
%   to input a non-negative value.  Re-prompt the user if necessary until the
%   user enters a non-negative value, at which point the entered value should
%   be rounded to the nearest integer and stored to replace the original
%   negative value.
```

Make sure that your code for `cleanData` does not use the following built-in functions: `max`, `min`, `sort`, and `find`. To demonstrate how this function works, suppose that `weeklyWeather = [34 -2 41 44]`. At some point in the execution of this function, the message "Input a non-negative temperature value for week 2: " should be printed on the Command Window. This can be achieved using the command

```
input(sprintf('Input a non-negative temperature value for week \%d: ', k));
```

where `k` is the week index with negative temperature. Observe that `sprintf` works the same way as `fprintf` in building a string, but `sprintf` returns a string (then used above as the argument to function `input`) while `fprintf` prints the string to the Command Window.

Again, test your function in the `assignment1a.m` file using your own input values for `weeklyWeather` wherein some of the data values are negative.

## 2 Self-check list

The following is a list of the minimum *necessary* criteria that your assignment must meet in order to be considered *satisfactory*. Failure to satisfy any of these conditions will result in an immediate request to resubmit your assignment. Save yourself and the graders time and effort by going over it before submitting your assignment for the first time.

Note that, although all of these are necessary, meeting all of them might still not be *sufficient* to consider your submission satisfactory. We cannot list everything that could be possibly wrong with any particular assignment!

- ∆ Comment your code! If any of your functions is not properly commented, regarding function purpose and input/output arguments, you will be asked to resubmit.
- ∆ Suppress all unnecessary output by placing semicolons (;) appropriately. At the same time, make sure that all output that your program intentionally produces is formatted in a user-friendly way.
- ∆ Make sure your functions' names are *exactly* the ones we have specified, *including* case.
- ∆ Check that the number and order of input and output arguments for each of the functions matches exactly the specifications we have given.
- ∆ Test each one of your functions independently, whenever possible, or write short scripts to test them.
- ∆ Check that your scripts do not crash (i.e., end unexpectedly with an error message) or run into infinite loops. Check this by running each script several times in a row. Before each test run, you should type the commands `clear all; close all;` to delete all variables in the workspace and close all figure windows.

## 3 Submission instructions

1. Upload files `generateWeatherData.m`, `isPerfect.m`, `findExtremeTemp.m`, `cleanData.m` and `assignment1a.m` to CMS in the submission area corresponding to Assignment 1a in CMS before the deadline.
2. Please do not make another submission until you have received and read the grader's comments.
3. Wait for the grader's comments and be patient.
4. Read and consider the grader's comments carefully.
5. If you need to resubmit, fix all the problems, *test all your code thoroughly*, and go back to Step 1. Otherwise you are done with this assignment. Well done!