Armstrong Atlantic State University
Engineering Studies
MATLAB Marina – Searching Exercises

1. Modify the `findEqual` function of Figure 1 to find only the first instance of a match rather than all of the matches. If no matches are found the function should still return an empty array. Test your modified function for the list [5, 4, 10, 2, 4, 6, 3, 7, 8, 1] and the keys 4, 3, and 13 (this will require three test calls, one for each of the values).

```
function result = findEqual(data, key)
% ------------------------------------------------------
% findEqual.m
% ------------------------------------------------------
% findEqual determines the location (indices) of all
% instances of the key in a 1D array
% ------------------------------------------------------
% Syntax: result = findEqual(data, key)
%  data is the 1D array of numbers
%  key is the number to search for
%  result is a array of indices where matches were found
% ------------------------------------------------------
% Notes: an empty array is returned if no matches
% ------------------------------------------------------
n = 0;
result = [];
% go through the array and save indices matches are found at
for k = 1:1:length(data)
    if (data(k) == key)
        n = n + 1;
        result(n) = k;
    end
end
end
```

Figure 1, `findEqual` function (Linear Search)

2. Modify the `findEqual` function of Figure 1 to search starting from the end of the list rather than start of the list. Test your modified function for the list [5, 4, 10, 2, 4, 6, 3, 7, 8, 1] and the values 4, 3, and 13. Is there any benefit to searching from back to front as opposed to front to back using a linear search?
3. Write a MATLAB function `findLessThan` that will take a 1D array of numbers and return the indices of all elements less than the key. The `findLessThan` function will be similar to the `findEqual` function of Figure 1. Test your `findLessThan` function for the list [5, 4, 10, 2, 4, 6, 3, 7, 8, 1] and the values 4, 0, and 13.

4. Write a MATLAB function `findStringinCellArray` that takes a 1D cell array of strings returns the indices of the cells containing the string key. The cell array is assumed to be depth one. Hints: for the comparison remember to extract the contents of the cell containers from the cell array and since strings are being compared use the MATLAB `strcmp` function rather than a logic equal comparison.
5. Write a MATLAB function `findStudentsByGPA` that takes a `student` structure array and a key and returns the indices of the `student` structure array corresponding to the students' whose GPA is greater than the key. The `student` structure consists of three fields: name (a string), hours (a number), GPA (a number in the range of 0 to 4).
6. Write a MATLAB program that will display the student names from a `student` structure array that are on track to graduate with honors. Assume a student graduates with honors if their GPA is greater than 3.5/4.0. Use the `findStudentsByGPA` function written for exercise 5 to identify the students meeting the criteria. The MATLAB code of Figure 2 will create a student structure array.

```
names = {'Bob Smith', 'Betty Jones', 'Jack Post', 'Ed Holt',
'Lisa Green', 'Ted Moss'};
creditHours = {43, 60, 21, 55, 94, 18};
GPAs = {3.1, 3.8, 2.3, 3.6, 2.7, 1.7};
studentArray = struct('name', names, 'hours', creditHours,
'GPA', GPAs);
```

Figure 2, MATLAB Code to Create `student` Structure Array

Last modified Wednesday, November 13, 2013