For the birds
Selecting and sorting data

CS112 Scientific Computation
Department of Computer Science
Wellesley College

A bird database

<table>
<thead>
<tr>
<th>Name</th>
<th>Family</th>
<th>Habitat</th>
<th>Size</th>
<th>Wingspan</th>
<th>BackColor</th>
<th>UnderColor</th>
<th>HeadColor</th>
<th>Spotted</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>GreatEgret</td>
<td>heron</td>
<td>marshes</td>
<td>39</td>
<td>51</td>
<td>white</td>
<td>white</td>
<td>white</td>
<td>no</td>
<td>longLegs</td>
</tr>
<tr>
<td>GreatBlueHeron</td>
<td>heron</td>
<td>marshes</td>
<td>48</td>
<td>72</td>
<td>blue/gray</td>
<td>black</td>
<td>black/white</td>
<td>no</td>
<td>longLegs</td>
</tr>
<tr>
<td>SnowyEgret</td>
<td>heron</td>
<td>marshes/ponds</td>
<td>24</td>
<td>38</td>
<td>white</td>
<td>white</td>
<td>white</td>
<td>no</td>
<td>longLegs</td>
</tr>
<tr>
<td>GreenHeron</td>
<td>heron</td>
<td>marshes/ponds</td>
<td>19</td>
<td>25</td>
<td>green</td>
<td>brown</td>
<td>black</td>
<td>no</td>
<td>darkBill</td>
</tr>
<tr>
<td>AmericanBittern</td>
<td>heron</td>
<td>marshes</td>
<td>28</td>
<td>45</td>
<td>brown</td>
<td>brown</td>
<td>brown</td>
<td>no</td>
<td>greenLegs</td>
</tr>
<tr>
<td>LittleEgret</td>
<td>heron</td>
<td>lagoons/ponds</td>
<td>62</td>
<td>96</td>
<td>white</td>
<td>white</td>
<td>white</td>
<td>no</td>
<td>orangeBill</td>
</tr>
<tr>
<td>CanadaGoose</td>
<td>waterfowl</td>
<td>lagoons/ponds</td>
<td>40</td>
<td>72</td>
<td>brown</td>
<td>brown</td>
<td>black/white</td>
<td>no</td>
<td>fliesInV</td>
</tr>
<tr>
<td>MuteSwan</td>
<td>waterfowl</td>
<td>lagoons/ponds</td>
<td>36</td>
<td>58</td>
<td>white</td>
<td>white</td>
<td>white</td>
<td>no</td>
<td>orangeBill</td>
</tr>
<tr>
<td>AmericanBlackDuck</td>
<td>waterfowl</td>
<td>wetlands</td>
<td>23</td>
<td>36</td>
<td>brown</td>
<td>brown</td>
<td>brown</td>
<td>no</td>
<td>yellowBill</td>
</tr>
<tr>
<td>Mallard</td>
<td>waterfowl</td>
<td>wetlands</td>
<td>24</td>
<td>36</td>
<td>brown</td>
<td>gray</td>
<td>green</td>
<td>no</td>
<td>purpleCheek</td>
</tr>
<tr>
<td>BlueWingedTeal</td>
<td>waterfowl</td>
<td>marshes/ponds</td>
<td>15</td>
<td>24</td>
<td>brown</td>
<td>brown</td>
<td>blue/gray</td>
<td>yes</td>
<td>blueShoulders</td>
</tr>
<tr>
<td>Osprey</td>
<td>hawk/eagle</td>
<td>wetlands</td>
<td>23</td>
<td>72</td>
<td>brown</td>
<td>brown</td>
<td>white</td>
<td>yes</td>
<td>whiteLegs</td>
</tr>
<tr>
<td>BaldEagle</td>
<td>hawk/eagle</td>
<td>wetlands</td>
<td>32</td>
<td>80</td>
<td>brown</td>
<td>brown</td>
<td>white</td>
<td>yes</td>
<td>yellowBill</td>
</tr>
<tr>
<td>NorthernHarrier</td>
<td>hawk/eagle</td>
<td>marshes/fields</td>
<td>22</td>
<td>54</td>
<td>gray</td>
<td>gray</td>
<td>brown</td>
<td>no</td>
<td>longTail</td>
</tr>
<tr>
<td>CooperHawk</td>
<td>hawk/eagle</td>
<td>wood</td>
<td>17</td>
<td>28</td>
<td>gray</td>
<td>gray</td>
<td>gray</td>
<td>no</td>
<td>orangeLegs</td>
</tr>
<tr>
<td>Redtail Hawk</td>
<td>hawk/eagle</td>
<td>woods/fields</td>
<td>22</td>
<td>58</td>
<td>brown</td>
<td>white</td>
<td>brown</td>
<td>yes</td>
<td>orangeTail</td>
</tr>
<tr>
<td>Peregrine Falcon</td>
<td>hawk/eagle</td>
<td>marshes/cities</td>
<td>18</td>
<td>44</td>
<td>gray</td>
<td>gray</td>
<td>black/white</td>
<td>no</td>
<td>sideburn</td>
</tr>
</tbody>
</table>

Suppose you’d like to select all the herons, select the birds with giant wingspans, sort by size or wingspan, sort alphabetically by name...
Let’s load it in and see what we have

% loadBirds.m
fid = fopen('birds.txt');
birds = textscan(fid, '%s %s %s %u %u %s %s %s %s', ...  
    'headerlines', 1);
fclose(fid);

function printBirdInfo (birds)
% prints out all the information stored in the input cell array
for i = 1:length(birds{1})
    disp(sprintf('%22s %10s %17s %3u %3u %12s %12s %12s %4s %17s', ...  
        birds{1}{i}, birds{2}{i}, birds{3}{i}, birds{4}{i}, birds{5}{i}, ...  
        birds{6}{i}, birds{7}{i}, birds{8}{i}, birds{9}{i}, birds{10}{i}))
end

Selecting all the herons

function herons = getHerons (birds)
% find the indices of all birds from the heron family
indices = find(strcmp(birds{2}, 'heron'));
% create an empty cell array and fill it with all the % information from the heron family
herons = cell(1,10);
for i = 1:10
    herons{i} = birds{i}(indices);
end

Exercise: select birds with large wingspans (> 48")
MATLAB sort function

```matlab
>> nums = [7 2 9 7 8 3 6 1 3 4];
>> sortNums = sort(nums)
sortNums =
    1 2 3 3 4 6 7 7 8 9
>> sortNums = sort(nums, 'descend')
sortNums =
    9 8 7 7 6 4 3 3 2 1
>> [sortNums sortIndices] = sort(nums, 'ascend')
sortNums =
    1 2 3 3 4 6 7 7 8 9
sortIndices =
    8 2 6 9 10 7 1 4 5 3
```

**Exercise:** What does the expression `nums(sortIndices)` return?

Now let’s sort a cell array of strings

```matlab
>> words = {'early' 'cloud' 'heights' 'a' 'black' 'great' 'from' 'descended'};
>> sortWords = sort(words)
sortWords =
    'a' 'black' 'cloud' 'descended' 'early' 'from' 'great' 'heights'
>> words = {'early' 'Cloud' 'heights' 'A' 'black' 'Great' 'from' 'Descended'};
>> [sortWords sortIndices] = sort(words)
sortWords =
    'A' 'Cloud' 'Descended' 'Great' 'black' 'early' 'from' 'heights'
sortIndices =
    4 2 8 6 5 1 7 3
```

Hmmmm.... What’s going on here???
Remember the ASCII code?

When comparing the order of two strings MATLAB uses the order of characters in the ASCII code in which all capital letters appear before all lowercase letters

**Exercise:** Write a function that sorts a cell array of words alphabetically, independent of capitalization

Sorting the bird data

```matlab
function sortedData = sortByWingspan(birds)

% sort the bird information by wingspan
[temp indices] = sort(birds{5});

% create an empty cell array and fill it with all the bird % information in sorted order
sortedData = cell(1,10);
for i = 1:10
    sortedData{i} = birds{i}(indices);
end
```

**Exercise:** sort the birds alphabetically by name
On the flip side...

Suppose we have some nice, orderly information that we want to
scramble

>> order = randperm(10)
order =
   6  2  5  1  4  8 10  3  7  9
>> conditions = [2.0, -2.0, 2.5, -2.5, 3.0, -3.0, 3.5, -3.5, 4.0, -4.0];
>> newConditions = conditions(order)
newConditions =
   -3.0  -2.0  3.0  2.0  -2.5  -3.5  -4.0  2.5  3.5  4.0

Exercise: write a function makeAnagram that has an input string and returns an
anagram of the string

Suppose the bird data is stored in an Excel spreadsheet...
... and we want to create the same **birds** cell array as before

```matlab
[numData txtData allData] = xlsread('birdInfo.xls');
birds = cell(1,10);
for col = 1:10
    if ((col == 4) | (col == 5))
        birds{1,col} = cell2mat(allData(2:60,col));
    else
        birds{1,col} = allData(2:60,col);
    end
end
```

(now we can use printBirdInfo, getHeron, and sortByWingspan as before)