Solutions to Homework 7

Help Center

Problem integerize

traditional solution with a single if-elseif-statement

```
function name = integerize(A)
    mx = max(A(:));
    name = 'NONE';
    if mx <= intmax('uint8')
        name = 'uint8';
    elseif mx <= intmax('uint16')
        name = 'uint16';
    elseif mx <= intmax('uint32')
        name = 'uint32';
    elseif mx < intmax('uint64')
        name = 'uint64';
    end
end
```

Problem integerize (alternative solution)

using a cell vector of strings and a for-loop instead

```
function cl = integerize(A)
    cls = {'uint8'; 'uint16'; 'uint32'; 'uint64'};
    cl = 'NONE';
    mx = max(A(:));
    for ii = 1:length(cls)
        if intmax(cls{ii}) >= mx
            cl = cls{ii};
            break;
        end
    end
end
```

Problem integerize (alternative solution)

using a cell vector of strings and vector indexing instead

```
function iclass = integerize(A)
    c = {'uint8','uint16','uint32','uint64','NONE'};
    % Array of maximum values for each class
    x = 2.^[8,16,32,64] - 1;
    % Index into names, based on size of largest element of A
```

```
    iclass = c{sum(max(A(:))>x)+1};
end
```

Problem May2015

```
function sub_May2015
    days = ['Thu'; 'Fri'; 'Sat'; 'Sun'; 'Mon'; 'Tue'; 'Wed' ];
    for ii = 1:31
        m(ii).month = 'May';
        m(ii).date = ii;
        m(ii).day = days(rem(ii,7)+1,:);  % +1 is needed because 0 is an invalid index
    end
end
```

Problem June2015

traditional solution with a for-loop

```
function m = June2015
    days = [ 'Sun'; 'Mon'; 'Tue'; 'Wed'; 'Thu'; 'Fri'; 'Sat'];
    for ii = 1:30
        m{ii,1} = 'June';
        m{ii,2} = ii;
        m{ii,3} = days(rem(ii,7)+1,:);
    end
end
```

Problem June2015 (alternative solution)

using MATLAB built-in functions instead

```
function x = June2015
    % Make vector of dates for June 2015
    t = datetime(2015,6,1:30);
    % Make a cell array from the components of t
    x = cat(1,month(t,'name'),num2cell(day(t)),day(t,'shortname'))';
end
```

Problem codeit

traditional solution, looking at one char at a time

```
function out = codeit(in)
    for ii = 1:length(in)
        if in(ii) <= 'z' && in(ii) >= 'a'       % lower case
            out(ii) = 'a' + 'z' - in(ii);       % encrypt it
        elseif in(ii) <= 'Z' && in(ii) >= 'A'   % upper case
```

```
            out(ii) = 'A' + 'Z' - in(ii);       % encrypt it
        else                                    % everything else
            out(ii) = in(ii);                   % no change needed
        end
    end
end
```

Problem codeit (alternative solution)

using logical indexing instead, the input and the output arguments are the same

```
function txt = codeit (txt)
    U = txt > 64 & txt < 91;    % identify uppercase
    L = txt > 96 & txt < 123;   % identify lowercase
    txt(U) = char(155-txt(U));  % encrypt uppercase
    txt(L) = char(219-txt(L));  % encrypt lowercase
end
```

Problem dial

translating the actual requirements straight to code works, but it is pretty long and somewhat awkward

```
function ph = dial(str)
    code = {'ABC'; 'DEF'; 'GHI'; 'JKL'; 'MNO'; 'PRS'; 'TUV'; 'WXY'};
    ph = str;                       % set the output to the input
    for ii = 1:length(str)
        c = str(ii);                % the current char from the input
        if c == ' ' || c == '-' || c == '(' || c == ')'
            ph(ii) = ' ';           % these characters need to turn into spaces
            continue;
        elseif (c >= '0' && c <= '9') || c == '#' || c == '*'
            continue;               % these need to remain unchanged
        else
            n = -1;
            for jj = 1:length(code)
               if ~isempty(strfind(code{jj},c))   % looking for legal letters
                   n = jj + 1;      % Found it! ABC on the dial maps to 2 not 1, hence
the +1
                   break;
               end
            end
            if n == -1              % if we did not find a valid letter
                ph = [];            % need to return []
                return;
            end
            ph(ii) = char('0' + n); % otherwise, add the char for the right number
        end
    end
end
```

Problem dial (alternative solution)

no loop and a single if-statement

```
function ph = dial(str)
    % the variable code has the characters' required mapping starting from space, endin
g with Y
    % x is for illegal input (e.g., see how Q maps to x in-between 7-s)
    code = ' xx#xxxx  *xx xx0123456789xxxxxxx2223334445556667x77888999';
    ph = [];        % default return value in case of illegal input
    n = str-' '+1;  % creates a vector of indexes into code from the input characters
    % the first two sum()-s  check for out-of-range input (smaller than space or larger
than Y )
    % the third sum() checks for any input char mapping to x, that is, illegal input
    if ~((sum(n <= 0) + sum(n > length(code))) || sum(code(n) == 'x'))
        ph = code(n);   % a single assignment does the actual transformation of the inp
ut string
    end
end
```

Problem replace

using a for-loop and logical indexing

```
function strs = replace(strs,c1,c2);
    for ii=1:length(strs)                  % for each string in the cell vector
        strs{ii}(strs{ii} == c1) = c2;     % replace all c1-s with c2-s at once
    end
end
```

Problem roman

problem size is small, so it is easier to simply enumerate all 20 numbers

```
function num = roman(rom)
    romans = { 'I' 'II' 'III' 'IV' 'V' 'VI' 'VII' 'VIII' 'IX' 'X' ...
        'XI' 'XII' 'XIII' 'XIV' 'XV' 'XVI' 'XVII' 'XVIII' 'XIX' 'XX'};
    num = uint8(0);
    for ii = 1:20
        if strcmp(rom,romans{ii})
            num = uint8(ii);
            break
        end
    end
end
```

Problem roman (alternative solution)

using find() instead of a loop

```
function ar = roman(str)
    allstr = {'I','II','III','IV','V','VI','VII','VIII','IX','X',...
        'XI','XII','XIII','XIV','XV','XVI','XVII','XVIII','XIX','XX'};
    ar = find(strcmp(allstr,str));  % find() returns the indexes of non-zero elements
    if isempty(ar)                  % if no match, input is bad
        ar = 0;                     % no need to convert to uint8 yet
    end
    ar = uint8(ar);                 % convert to uint8
end
```

Problem censor

```
function out = sub_censor(strs,str)
    out = {};                               % creates the output from scratch
    for ii = 1:length(strs)                 % for each string in the cell vector
        if isempty(strfind(strs{ii},str))   % if the substring is not found
            out = [out strs{ii}];           % the current string goes into the output
        end
    end
end
```

Published with MATLAB® R2014a

Created Tue 9 Jun 2015 8:02 AM PDT

Last Modified Tue 9 Jun 2015 8:43 AM PDT