Question 1: (30 points)

Part (a): (2 points)

What does vector `v` look like after the following script is executed?

```
v = [0 1];
for k = 1:3
    v = [1 v];
end
```

Before: 0 1

After: **1 1 1 0 1**

Part (b): (2 points)

What does vector `w` look like after the following script is executed?

```
w= [3 2 1];
w(w(3)) = w(1);
```

Before: 3 2 1

After: **3 2 1**

Part (c): (10 points)

Assume that `a` and `b` are initialized scalars with `a<b`. Consider the following code fragment:

```
x= linspace(a,b,n);
y= sin(x);
```

Write an equivalent fragment that does not use function `linspace` and only calls the sine function with scalar input values.

```
h= (b-a)/(n-1);
for k= 1:n
    x(k)= a + (k-1)*h;
    y(k)= sin(x(k));
end
```

Question 1 continues on next page

Question 1, continued

Part (d): (6 points)

Assume that `score` is an initialized vector containing integer values in the interval [0,100]. (For example, `score` is a vector of student scores on a test). Write one statement on the blank below to complete the code fragment for drawing a histogram of the scores (with one bar for each score values 0, 1, 2, …, 100).

```
count= zeros(1,101);  % count will be used to store the histogram data
for k= 1:length(score)
    count(score(k)+1)= count(score(k)+1) + 1;
end
bar(0:100, count)  % draw a histogram of the scores
```

Part (e): (5 points)

Given the following function:

```
function f = evaluateQuadratic(a,b,c,x)
f= a*(x^2) + b*x + c;
```

What is the output when the following script is executed?

```
a=1;  b=-1;  c=3;  x=2;
f= evaluateQuadratic(c,b,a,x)
```

Output:

f =

11

Part (f): (5 points)

Given the following function:

```
function y = flip(x)

n= length(x);
for k= 1:n
    x(n-k+1)= x(k);
end
y= x;
```

What is the output when the following script is executed?

```
y= [10 20 30 40];
y= flip(y)
```

Output:

y =

10 20 20 10

Question 2: (20 points)

Write a function `s2hms` to convert a time in seconds to a time in hours, minutes, and seconds. The function has one parameter (`sec`) and returns three numbers: `h`, `m`, and `s`. Read the given function comment below; write the function header and the function body.

```
function [h, m, s] = s2hms(sec)
```

```
% Convert a time expressed in seconds (sec) to the number of hours (h),
% minutes (m), and seconds (s).  h and m are integer values and
% 0<=m,s<60.  Assume sec>=0.
```

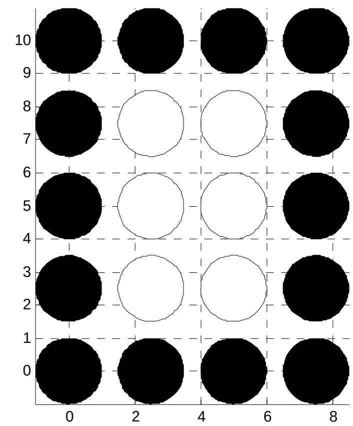
```
h= floor(sec/3600);
sec= sec - h*3600;  % OR:  sec= rem(sec,3600)
m= floor(sec/60);
s= sec - m*60;
```

Assume function `s2hms` has been written correctly. Write a script to print the number of times in a day that h>m>s. Check whole seconds from 0 to 60×60×24-1. You must use function `s2hms` to solve this problem.

```
maxSeconds= 60*60*24-1;

count= 0;  % No. of times when h>m>s

for k= 0:maxSeconds

    [h, m, s]= s2hms(k);

    if (h>m && m>s)

        count= count + 1;
    end
end

disp(sprintf('h>m>s %d times a day', count))
```

Question 3: (25 points)

Complete function `drawFrame` below to draw a "frame" made up of black and white disks. Each disk is of unit radius and the lower left disk is centered at (0,0). Shown on the right is an example of a 5-by-4 frame with a spacing of 0.5 between disks. The function call to produce this example is `drawFrame(5,4,0.5)`.



Assume that function DrawDisk is available. To draw a black disk of unit radius at position (3,4): `DrawDisk(3,4,1,'k')`

Write only the code to draw the disks. The grid lines are provided for your convenience—you do not need to draw them.

```
function drawFrame(h,w,s)
% Draw a frame composed of h-by-w black and white disks of unit radius
% with space s between the disks.  Black disks form the border; white
% disks are in inside.  The lower left disk is centered at (0,0).
% Assume h,w>2 and s>=0.

axis equal
hold on
```

```
d= 2+s;  % distance from center to center

for y= 0 : d : (h-1)*d
    for x= 0 : d : (w-1)*d
        if (x==0 || x==(w-1)*d || y==0 || y==(h-1)*d)  % border
            DrawDisk(x,y,1,'k')
        else
            DrawDisk(x,y,1,'w')
        end
    end
end
```

```
% An alternative

d= 2+s;  % distance from center to center

for r= 1:h
    y= (r-1)*d;
    for c= 1:w
        x= (c-1)*d;
        if (r==1 || r==h || c==1 || c==w)  % border
            DrawDisk(x,y,1,'k')
        else
            DrawDisk(x,y,1,'w')
        end
    end
end
```

```
hold off
```

Question 4: (25 points)

Complete function `findPrefix(p,s)` below to return the position of the first occurrence of a word that begins with string `p` in string `s`. If no word in `s` begins with string `p`, the function returns -1. For full credit, your algorithm should be efficient—stop after the first occurrence has been found. The only built-in functions that you may use are `length` and `strcmp`. Assume that `p` contains only lower case letters and `s` contains lower case letters and blanks. Below are four examples:

p	s	*Returned value*
mat	there is a mat in the lab	12
mat	there is a bat in the lab	-1
mat	matt uses matlab on a mat	1
mat	format a plot in matlab	18

```
1234567891111111111222222
         0123456789012345
```

In the last example above, the word “format” in `s` includes the substring ‘mat’ but that doesn’t count since ‘mat’ does not appear in the beginning of the word.

```
function k = findPrefix(p, s)
% k is the position in string s of the first occurrence of a word that
%   begins with string p
% k is -1 if no word in string s begins with string p
% p contains lower case letters only
% s contains lower case letters and blanks only
```

```
len= length(p);  % the length of the word pattern

s= [' ' s];  % Pad s with a leading space
k= 2;        % current index in s to start checking
found= 0;
% While prefix p is not found, check every substring s(k:k+len-1) against p
while k<=length(s)-len+1 && ~found
    if s(k-1)==' '  % only need to look for p if a blank is at s(k-1)
        found= strcmp(s(k:k+len-1), p);
    end
    k= k+1;
end
if ~found    % OR: if found==0
    k= -1;
else
    k= k-2;  % need -1 because in loop body k incremented after comparison
             % need another -1 because s was padded with a leading space
end
```

```
pat= [' ' p];       % the word pattern to look for
len= length(pat);  % the length of the word pattern

% Pad string s with a leading blank
s= [' ' s];
k= 1;
% While prefix p is not found, check every substring s(k:k+len-1) against pat
while k<=length(s)-len+1 && strcmp(s(k:k+len-1), pat)~=1
    k= k+1;
end
if k>length(s)-len+1  % k exceeds possible starting index, so prefix not found
    k= -1;
end
% If found, k needs no adjustment since both p and s were padded with an
% extra leading blank.
```

```
pat= [' ' p];       % the word pattern to look for
len= length(pat);  % the length of the word pattern

% Pad string s with a leading blank
s= [' ' s];
k= 1;
found= 0;
% While prefix p is not found, check every substring s(k:k+len-1) against pat
while k<=length(s)-len+1 && ~found
    found= strcmp(s(k:k+len-1), pat);
    k= k+1;
end
if found     % OR: if found==1
    k= k-1;  % need -1 because in loop body k incremented after comparison
else
    k= -1;
end
```

```
% Check first word in s
if strcmp(s(1:length(p)), p)
    k= 1;
    return
end

% Check all of s
pat= [' ' p];       % the word pattern to look for
len= length(pat);  % the length of the word pattern

k= 1;  % ok to start at 2 (assume s starts with a letter)
found= 0;
% While prefix p is not found, check every substring s(k:k+len-1) against pat
while k<=length(s)-len+1 && ~found
    found= strcmp(s(k:k+len-1), pat);
    k= k+1;
end
if ~found     % OR: if found==0
    k= -1;
end
% If found, k needs no adjustment:
%   The 1st char in pat is the padded blank, so need to add 1 to k, but
%   an extra 1 was added already since in the loop body k is incremented
%   after the comparison.
```