Some old exam questions

1. A Pascal's triangle with levels 0 to 4 is shown below. Level 0 has a single value, and each value on subsequent levels is the sum of the two entries diagonally above in the previous level of the triangle. For example, the value 6 in level 4 is the sum of the values 3 and 3 in level 3.

1	*level 0*
1 1	*level 1*
1 2 1	*level 2*
1 3 3 1	*level 3*
1 4 6 4 1	*level 4*

Complete function `pascalVector` below to return the row vector corresponding to a specified level of Pascal's triangle. For example, if level `lev` is 4, then the returned vector must be [1, 4, 6, 4, 1]. Assume that `lev` is a non-negative integer. The only Matlab built-in functions allowed are `zeros`, `ones`, and `length`. Do not use the formula for binomial coefficients to solve this problem. Use a loop (or loops): the vector for each level is based on the vector from the previous level.

```
function p = pascalVector(lev)
% p is the vector corresponding to level lev of Pascals triangle
```

2. Complete the following function.

```
function MyHistogram(v)
% Draw a histogram for the data in v using asterisks in the COMMAND WINDOW (not figure window).
% v is a vector of non-negative values.
% The histogram is scaled so that the largest data value is represented by
% ten asterisks. Round as necessary in order to draw whole asterisks.
% Example: v = [12 4.1 0.5 9.2 20]
% Output in Command Window:
% ******
% **
%
% *****
% **********
```

3. **(a)** Implement function `isIn` as specified. *The only built-in function that you may use is* `length`.

```
function alfa = isIn(x, v)
% alfa is 1 if value x is in vector v.  Otherwise alfa is 0.
% x is an integer.  v is a vector of integers, possibly of length 0.
% If v has length 0 (v is the empty vector), then alfa is 0.
```

(b) Let `a` and `b` be non-empty vectors of integers. We define the intersection set of `a` and `b` to be the distinct values that appear in *both* vectors `a` and `b`. For example, if

```
a = [4 2 2 5 3 8 6]
b = [3 5 1 6 4 5 5 0 7]
```

then the intersection set of `a` and `b` is the vector `[4 5 3 6]` (the order of the values in the vector does not matter). Implement function `intersectionSet` as specified, making effective use of function `isIn`. *The only built-in function that you may use is* `length`.

```
function s = intersectionSet(a,b)
% a and b are vectors of integers.  a and b are not empty.
% Vector s contains only the values that are in both vectors a and b.
% Vector s contains distinct values.  s may be empty.
```

4. Implement function `partialStrings` as specified. *Do not use vectorized code.*

```
function M = partialStrings(CA, t)
% CA is a length n cell array of "strings"; a "string" is a row char vector.
% t is a positive integer.
% M is an n-by-t matrix of characters:
%   Row r of M contains the first t characters of the rth char vector in CA.
%   If the rth char vector in CA is shorter than t, pad the char vector with '!'.
% NO VECTORIZED CODE
```

For example, let cell array `CA` hold these five char vectors:

```
'd'   'a2c'   '# _!.a'   ''   'ap'
```

Then the function call `M = partialStrings(CA,3)` should return a matrix `M` like this:

```
['d!!' ; ...
 'a2c' ; ...
 '# _' ; ...
 '!!!' ; ...
 'ap!']
```