**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`.

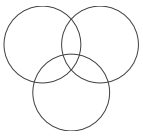
```
function s = intersectionSet(a,b)
% a and b are vectors of integers.  a and b are not empty.
% Vector s contains only the values that are in both vectors a and b.
% Vector s contains distinct values.  s may be empty.
```

4. **(a)** Implement this function:

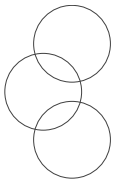
```
function z = overlap(diskA, diskB)
% z is 1 (true) if diskA and diskB overlap; otherwise z is 0 (false).
% diskA and diskB are each a disk structure with the following fields:
%   x: x-coordinate of center of disk
%   y: y-coordinate of center of disk
%   radius: radius of disk
```

**(b)** Implement the following function to return the indices of disk triplets that overlap. Three disks form a triplet if every disk overlaps with each of the other two. Make effective use of function `overlap` from part (a). Your code should be efficient—avoid unnecessary iterations.

```
function idx = diskTriplets(D)
% D is a 1-d array of disk structures; each structure has fields as defined in
%   part (a). Assume D has a length greater than 3.
% idx is a vector of indices indicating all triplet overlap combinations. For example,
%   if disks 2, 4, and 5 form a triplet and disks 3, 4, and 6 form a triplet, idx
%   should be the vector [2 4 5 3 4 6].  Other orderings of triplets are acceptable,
%   however each triplet should only appear once.
```



A triplet



Not a triplet