CS100M Section Exercise 7

1 Examining a subarray

[From Lab Exercise 6] Write a function `vectorQuery(v,n,r)` to determine whether the number `r` appears in the first `n` cells of vector `v`. The function returns 1 if `r` is in the first `n` cells of `v` and 0 otherwise. Your function assumes that `v` is a vector of numbers, `n` is a positive integer, and `r` is a number. *Use a loop* to do the search. Make sure that the loop index doesn't go "out of bounds" (if `n` is greater than the length of vector `v`).

2 Concatenating arrays

[From Lab Exercise 6] Write a function `sequence(m)` that generates a sequence of random *integer* numbers between 1 and m, inclusive, stopping when a value is repeated for the first time. The function returns an array containing all the numbers generated (in the order in which they were generated) except for the last value that is a repeated occurrence.

Example: If the generated sequence is 3 1 9 5 7 2 5, the array to be returned should be 3 1 9 5 7 2.

Hints: 1) Use the function `vectorQuery` that you have developed already. 2) The symbol for the empty array is `[]`. When "building" an array, the space or comma separator puts two items side by side—creates a row. Below is an example:

```
v= [];       % v is empty array
v= [v 7];    % now v is [7]
v= [v -5];   % now v is [7, -5]
```

3 Reverse complement

In the DNA double helix, two strands twist together and "face" each other. The two strands are reverse-complementary, i.e., reading one strand in reverse order and exchanging each base with its complement gives the other strand. A and T are complementary; C and G are complementary.

For example, given the DNA sequence

```
AGTAGCAT
```

the reverse sequence is

```
TACGATGA
```

so the reverse complement is

```
ATGCTACT
```

Write a function `rComplement(dna)` to return the reverse complement of a DNA strand. *Use a loop* to reverse the strand. Assume that `dna` contains only the letters 'A', 'T', 'C', and 'G'.

4 Counting a DNA pattern

Write a function `countPattern(dna,p)` to find out (and return) how many times a pattern `p` occurs in `dna`. Assume both parameters to be strings that contain the letters 'A', 'T', 'C', and 'G' only. Note that if `p` is longer than `dna`, then `p` appears in `dna` zero times. You may, but you don't need to, use the built-in function `strcmp` to compare two strings.