Problem 1

Write a function called `make_grid` that takes two inputs, `r` and `c`, and returns a 2-D list with `r` rows and `c` columns that is populated by '-' characters. This represents a population where all the cells are 'dead'.

```
make_grid(3,3) -> [ ['-','-','-'],
                    ['-','-','-'],
                    ['-','-','-'] ]
```

Hint: this *can* (but doesn't have to) be done with a single of code using list comprehensions!

```
assert make_grid(1,1) == [['-']]
assert make_grid(1,10) == [['-', '-', '-', '-', '-', '-', '-', '-', '-', '-']]
assert make_grid(10,1) == [['-'], ['-'], ['-'], ['-'], ['-'], ['-'], ['-'], ['-'], ['-'],
['-']]
assert make_grid(4,7) == [['-', '-', '-', '-', '-', '-', '-'], ['-', '-', '-', '-', '-',
'-', '-'], ['-', '-', '-', '-', '-', '-', '-'], ['-', '-', '-', '-', '-', '-', '-']]

g = make_grid(3,3)
g[0][1] = '*'
assert g == [['-', '*', '-'], ['-', '-', '-'], ['-', '-', '-']]
```

Problem 2

Write a function called `to_string` that takes a 2-D list `grid` as input and returns the values of all the characters in the grid as a single string with newline characters (`\n`) at the ends of each row.

```
assert to_string(make_grid(1,1)) == '-\n'
assert to_string(make_grid(1,10)) == '----------\n'
assert to_string(make_grid(10,1)) == '-\n-\n-\n-\n-\n-\n-\n-\n-\n-\n'
assert to_string(make_grid(4,7)) == '-------\n-------\n-------\n-------\n'
```

Problem 3

Write a function called `num_neighbors` that takes three inputs: `grid`, a 2-D list, `r` and `c`, the row and column of a cell. This function returns the number of neighbors of this cell that are 'alive'.

```
g = make_grid(5,5)
g[0][2] = '*'
g[0][3] = '*'
g[1][0] = '*'
g[1][2] = '*'
g[1][3] = '*'
g[2][2] = '*'
g[3][0] = '*'
g[4][0] = '*'
g[4][4] = '*'

assert num_neighbors(g,0,0) == 1
assert num_neighbors(g,0,3) == 3
assert num_neighbors(g,1,1) == 4
assert num_neighbors(g,2,2) == 2
assert num_neighbors(g,4,2) == 0
```