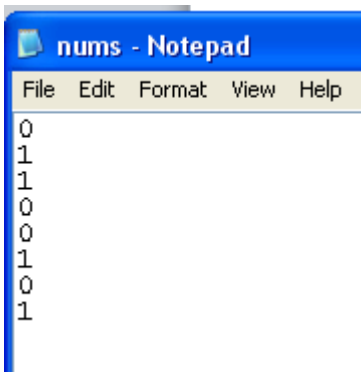
CS 1301 Recitation Assignment

You will be writing two functions. Save these functions to a file named `RA-March1.py`.

Function 1: Generating a List of Integers from a File

Function Description

Write a function named *intList* that takes in a single parameter which will be a string. This parameter will be the file name containing a sequence of numbers for you to construct a list from. You can assume that the file will always be formatted with a single integer per line as shown below:



Your function will read in the entire text file, and construct a list of integers. Each entry in the list will correspond to its respective position in the text file. For the sample case above, your constructed list will look like

```
[0, 1, 1, 0, 0, 1, 0, 1]
```

Once you construct the list of integers, your function should return the list.

Sample Function Call

Assume `nums.txt` exists as shown above.

```
>>> intList("nums.txt")
[0, 1, 1, 0, 0, 1, 0, 1]
```

Function 2: Prefix-Sum using Addition

Function Background

The prefix-sum operation (using addition as a binary operator) is an operation where each element in the list is the sum of all previous elements in the list. Another way of thinking about it is that for each index *k* in a list *L*,

$$L[k] = L[0] + L[1] + \ldots + L[K-1]$$

This algorithm can be sequentially implemented with a single for loop: simply iterate over a list (starting at index 1) and increment the value at each index by the value at the index before it.

A couple of examples are as follows:

```
Input:     [0, 1, 1, 0, 1, 1, 1]
Output:    [0, 1, 2, 2, 3, 4, 5]

Input:     [1, 1, 1, 1, 1, 1, 1]
Output:    [1, 2, 3, 4, 5, 6, 7]

Input:     [1, 4, 3, 0, 0, 1, 0]
Output:    [1, 5, 8, 8, 8, 9, 9]
```

At a broader level, the prefix-sum operation can be used in sequential and parallel algorithms for a variety of uses:

- To lexically compare strings of characters. For example, to determine that "strategy" should appear before "stratification" in a dictionary
- To evaluate polynomials
- To implement various sorting algorithms
- Many, Many More!

While we will not be discussing additional uses of the prefix-sum in this class, it is nice to have motivating examples to guide the utility of the function you are about to write.

Function Description

Your task is to write a function named *prefixSum*. The function should take in a single parameter: a list of integers.

Now, you need to calculate the prefix sum of this list (be sure to store the list with the prefix-sum calculations to a new variable).

Finally, your function should return a 2-tuple: The first element is the un-modified input list; the second element is the calculated prefix-sum of the constructed list.

Sample Function Call

```
>>> prefixSum([0, 1, 1, 0, 0, 1, 0, 1])
([0, 1, 1, 0, 0, 1, 0, 1], [0, 1, 2, 2, 2, 3, 3, 4])
```