Alex Lundin
Assignment 1

1.)

```
input (BinaryString, index)
output (numberOfOnes)

BinaryRecursion (BinaryString, index, numberOfOnes){

stringLength = BinaryString.length

if(stringLength < 1){

stringCharacter = charAtIndex (index)
index + 1
BinaryString = subString(index, stringLength)

if(stringCharacter equals "1"){
numberOfOnes + 1
}

BinaryRecursion (BinaryString, index, numberOfOnes)
}

else {
return numberOfOnes
}

}
```

2.)

a.)

[1 / (4^0)] + [1 / (4^1)] + [1 / (4^2)] + [1 / (4^3)] + ... +

[1 / 1] + [1 / 4] + [1 / 16] + [1 / 64] + ... +

Values approach but won't reach 2

b.)

[0 / (4^0)] + [1 / (4^1)] + [2 / (4^2)] + [3 / (4^3)] + ... +

[0 / 1] + [1 / 4] + [2 / 16] + [3 / 64] + ... +

Values approach but won't reach 1

3.) Fibonacci numbers

4.)
Prove:

(1)^3 + (2)^3 + (3)^3 + ... + (n)^3 = { (n^2) * [(n + 1)^2] }/ 4

Base Case:

Pick some small, trivial, usually degenerate values to start process
Base cases start at minimal value

N = 1

(1)^3 = { (1^2) * [(1 + 1)^2] }/ 4

1 = 1

Inductive Hypothesis:

Theorem assumed true up until some value, usually k + 1

1.) Statement with k

(1)^3 + (2)^3 + (3)^3 + ... (k)^3 = { ((k)^2) * [((k+ 1)^2] }/ 4

2.) Proof of k + 1 by substitution of the right hand value of above equation for k

(1)^3 + (2)^3 + (3)^3 + ... (k)^3 + (k+1)^3 = { ((k+1)^2) * [((k+1) + 1)^2] }/ 4

{ { ((k)^2) * [((k + 1)^2] }/ 4 } + (k+1)^3 = // substitute statement with k

{ (k^2)*[(k + 1)^2] + 4[(k+1)^3] } / 4 = // common denominators to combine into one fraction

{ [(k + 1)^2] * [(k^2) + 4[(k+1)^3]] } / 4 = // factor out (k + 1)^ 2

{ [(k + 1)^2] * [(k^2) + 4k + 4] } / 4 = // foil inside

{ [(k + 1)^2] * [(k + 2)^2] } / 4 = // reverse foil inside

{ [(k + 1)^2] * [(k + 2)^2] } / 4 = // original statement, with k + 1 substituted in for k

{ [(k + 1)^2] * [((k + 1) + 1)^2] } / 4 = // expanded the squared term to make it clean, this is
// the exact original from the first line of this section