# CS151: Data Structures - Prelab 1

James Capuder, Oberlin College February 8, 2015

## Return to the Pyramids

### 1: Given a height N, give the loops needed to print a pyramid of that height.

```
#Pyramid Generator in python.
n = eval(input("Please enter the desired height of the pyramid: "))
max = (n*2)-1
for i in range(n):
    stars = 1+(i*2)
    spaces = (max-stars)/2
    print(spaces*" " + stars*"*" + spaces*" ")
```

## HiLo Guessing Game

you and your friend (the computer) will play a guessing game. One of you will pick a number between 1 and 1000. The other one will guess numbers and the other will tell them if they guessed correctly or were too high/low.

### 2: Describe the logic needed to ensure that a user types in a guess between 1 and 1000.

To ensure that a user types in a guess between 1 and 1000, a try/catch statment could be implemented. First, since there will be multiple guesses, a userGuess function should be constructed, to be called every time a guess is required from the user. In this function, there would be a try/catch statement. Under "try," the the user would be prompted to enter their guess, and then an if/else statement would check to see if this input is between 1 and 1000. If so, the function would return the guess. If not, an assertion error would be raised and under "except," the user would be told that their input must be between 1 and 1000, and the userGuess function would be called again.

### 3: Describe an effective strategy to win this game if you are making the guesses.

Since you will always be told if your guess is above or below the chosen number, an effective strategy to win the game would be to guess a number in the center of the range in which you know the chosen number must be. For example, a good first guess would be 500. The computer would tell you if 500 is too high or too low, and based on this information, good follow-up guesses could be 250 or 750. In this example, lets say the computer tells you 500 is too low. You would then know the number is between

500 and 1000, so you could guess 750. The computer tells you that 750 is too high, so you could guess 625. You could follow this pattern until you finally hone in on the correct answer. This method provides you with tangible ranges which you know must contain the number, and thereby minimizes the number of guesses you must make.

## Redaction

### 4: Describe how you would accurately determine the number of lines in a text file.

First, a variable, lineCount, could be initialized as an integer with a value of 1. Then, a loop could iterate over every character in the file. In this loop, there could be an if/else statement. If a forward slash is encountered, the character at the next index would be checked. If it is "n," 1 would be added to lineCount, otherwise the loop would continue. This would provide the accurate number of lines in a text file.

### 5: Describe the data structures (i.e, the classes you will use) you would need to preform this calculation

This method for determining the number of lines in a text file would utilize the classes associated with strings, text files and the reading thereof.

## Benford's Law

Take a look at the Wikipedia entry for Benford's Law. We are going to compute the frequency distribution of digits in some files and see if this distribution (Benford's law) actually holds for them. As part of this, we will be drawing a simple text-based histogram chart based on the frequency of the digits

### 6: Describe the data structures you would need to perform this calculation.

An array of length 10 would be used to keep track of the counts of leading digits 1-9, with the value of the final index being the total count of words. Then another array of length 9 would be used to keep track of the proportions of each digit, so each index in the second array would have the value of $\frac{\text{Corresponding Digit}}{\text{Total Digits}}$.

### 7: Assuming that you have a list of integers representing counts and an integer N representing the height of the tallest bar, give an algorithm that would allow you to come up with an integer heigh of all the bars such that they are between 0 and N units tall and an appropriate proportion of the height of the tallest bar.

The proportion $\frac{\text{Highest Count in List}}{N}$, let's call it heightUnit, would provide you with the number of units representing a count of 1. Each integer in the given list would then be multiplied by heightUnit, and bars would be drawn with a height of each of the products. This ensures that no height would be greater than N, and that all the bars would be appropriately proportioned.