In this workshop, you'll write a program to produce a scalable word art tree. Your workshop leader will guide you through the process. Please ***do not*** *work ahead.*

```
           pp
          yppy
         typpyt
        htyppyth
       ohtyppytho
      nohtyppython
     pnohtyppythonp
    ypnohtyppythonpy
   typnohtyppythonpyt
  htypnohtyppythonpyth
 ohtypnohtyppythonpytho
nohtypnohtyppythonpython
           oo
pythnpythnpythnpythnpythn
```

1 Function warmups

1. Write a function, called `sumEven`, that given an integer N, returns the sum of all the even numbers from 1 up to and including N.

2. Write a program that counts the number of heads seen in `N` coin flips. The program should ask for `N` then call a function `countHeads` (which you write), that given a number N, flips a coin N times and returns the total number of heads seen.

2 Word Art Tree

3. Write a function called `getVowels` that given a string, returns a new string with all the vowels in the given string. The new string should include repeats.

4. Write a function called `getNonVowels` that given a string, returns a new string with all the non vowels in the given string. The new string should include repeats.

5. Write a function called `reverseString` that given a string, returns the reverse of that string.

6. Write a function called `mirrorString` that given a string returns a new string that is the given string mirrored. For example, given ‘hello’, the string returned would be ‘helloolleh’.

7. Write a function called `centerString` that given a string and a width, returns a string with the original string centered in that width. You may assume that the width is longer than the length of string s.

8. Write a function called `strToWidth` that given a string and a width, returns a string with the original string repeated to fill the width. If the number of letters in the string will not allow the length of the string to equal the width exactly, the returned string should overflow the width, not underflow it.

9. Write a function `makeGround`, that given a string and a width, and using only the consonants from the string, creates a string that will be the ground under the tree. The consonants should repeat enough times to span the entire width of the tree. The function should return the string.

10. Write a function `makeTrunk`, that given a string and a width, uses only the vowels in the string to create a trunk for the tree. The trunk should be centered on the width and contain the number of vowels doubled in a mirrored arrangement. The function should return a string that represents the trunk of the tree. The following is an example show the resulting trunk for a word with more vowels.

```
             pp
            appa
           nappan
          cnappanc
         acnappanca
        kacnappancak
       ekacnappancake
      pekacnappancakep
     apekacnappancakepa
    napekacnappancakepan
   cnapekacnappancakepanc
  acnapekacnappancakepanca
 kacnapekacnappancakepancak
ekacnapekacnappancakepancake
           aaeeaa
           aaeeaa
           aaeeaa
pnckpnckpnckpnckpnckpnckpnck
```

11. Write a function `makeLeaves`, that given a string and a scale, returns a string representing the branches and leaves of the tree. In the left half of the tree the string should appear backwards, in the right half the string should appear forwards.

12. Write a `main` function that asks the user for an input word and an input multiplier. The program should then print out the word art tree using the word. The height of the leaves of the tree should be the length of the word times the multiplier.

13. Modify your program (or identify how you would modify your program) so that a tree cannot grow if there is no ground, and have the program display a similar message so the user knows what happened.

14. Continue to add to your program as time allows. Be creative!