1. For the following Python program:

   ```python
   n = 4
   for i in range(0,n):
       s = ''
       for j in range(0,i):
           s = s + 'X'
       print(s)
   ```

   (a) [4 pts.] What does it print?
   (b) [6 pts.] What does each line do (provide a sentence for each line)?

2. [5 pts.] Provide a big-O complexity for the above code in terms of the value of the variable \( n \), as defined in the lecture notes on program correctness and complexity. (It may help to think about how many ‘X’s would be added if \( n \) were 100 or 1000 or 1,000,000 . . . )

3. PROGRAMMING: Write a single Python program — not a Makefile — to do all of the items below. Your code should be as short as possible.

   (Remember to include a short representative sample of input and output for all programming problems, as described in the previous problem set.)

   (a) [10 pts.] Read an FSA from standard input using the appropriate code from the ‘fsarec.py’ program in the lecture notes on file access using Python; then print that FSA to standard output in the same format (so that the output would be readable as input by the same program).

   (b) [10 pts.] Then reverse that FSA as defined in the lecture notes on regular expressions, and print the reversed FSA to standard output. (Hint: it may be easier to implement \( M \) as a list of \([q1,x,q2]\) lists.) Show the result of using your reversed FSA in fsarec.py.

For example, given the following via standard input:

```
S qHappy
F qHungry
M qHappy voc qHungry
M qHappy sil qHappy
M qHungry voc qHungry
M qHungry sil qHungry
```

your program should print:
4. PROGRAMMING: Write a single Python program — not a Makefile — to do all of the items below. Your code should be as short as possible.

(Remember to include a short representative sample of input and output for all programming problems, as described in the previous problem set.)

(a) [10 pts.] Read text from standard input, delete all punctuation marks, and print the word tokens with one on each line (delimit word tokens with spaces in the input).

(b) [10 pts.] Print the number of times each word appears in the input text.

(c) [10 pts.] Print the number of times each pair of adjacent words occurs in this input (even across sentences or paragraphs).

For example, given the following via standard input:

Lions, tigers and bears!

your program should print:

a)
Lions
tigers
and
bears

b)
Lions 1
tigers 1
and 1
bears 1

c)
Lions tigers 1
tigers and 1
and bears 1