8.7 Exercises

1. Write a program that asks the user to enter some text and then counts how many articles are in the text. Articles are the words `'a'`, `'an'`, and `'the'`.

2. Write a program that allows the user to enter five numbers (read as strings). Create a string that consists of the user's numbers separated by plus signs. For instance, if the user enters 2, 5, 11, 33, and 55, then the string should be `'2+5+11+33+55'`.

3. (a) Ask the user to enter a sentence and print out the third word of the sentence.
   (b) Ask the user to enter a sentence and print out every third word of the sentence.

4. (a) Write a program that asks the user to enter a sentence and then randomly rearranges the words of the sentence. Don't worry about getting punctuation or capitalization correct.
   (b) Do the above problem, but now make sure that the sentence starts with a capital, that the original first word is not capitalized if it comes in the middle of the sentence, and that the period is in the right place.

5. Write a simple quote-of-the-day program. The program should contain a list of quotes, and when the user runs the program, a randomly selected quote should be printed.

6. Write a simple lottery drawing program. The lottery drawing should consist of six different numbers between 1 and 48.

7. Write a program that estimates the average number of drawings it takes before the user's numbers are picked in a lottery that consists of correctly picking six different numbers that are between 1 and 10. To do this, run a loop 1000 times that randomly generates a set of user numbers and simulates drawings until the user's numbers are drawn. Find the average number of drawings needed over the 1000 times the loop runs.

8. Write a program that simulates drawing names out of a hat. In this drawing, the number of hat entries each person gets may vary. Allow the user to input a list of names and a list of how many entries each person has in the drawing, and print out who wins the drawing.

9. Write a simple quiz game that has a list of ten questions and a list of answers to those questions. The game should give the player four randomly selected questions to answer. It should ask the questions one-by-one, and tell the player whether they got the question right or wrong. At the end it should print out how many out of four they got right.

10. Write a censoring program. Allow the user to enter some text and your program should print out the text with all the curse words starred out. The number of stars should match the length of the curse word. For the purposes of this program, just use the"curse" words *darn*, *dang*, *freakin*, *heck*, and *shoot*. Sample output is below:

```
Enter some text: Oh shoot, I thought I had the dang problem
figured out. Darn it. Oh well, it was a heck of a freakin try.

Oh *****, I thought I had the **** problem figured out.
**** it. Oh well, it was a **** of a ****** try.
```

11. Section 8.3 described how to use the `shuffle` method to create a random anagram of a string. Use the `choice` method to create a random anagram of a string.

12. Write a program that gets a string from the user containing a potential telephone number. The program should print `Valid` if it decides the phone number is a real phone number, and `Invalid` otherwise. A phone number is considered valid as long as it is written in the form *abc-def-hijk* or *1-abc-def-hijk*. The dashes must be included, the phone number should contain only numbers and dashes, and the number of digits in each group must be correct. Test your program with the output shown below.

```
Enter a phone number: 1-301-447-5820
Valid
Enter a phone number: 301-447-5820
Valid
Enter a phone number: 301-4477-5820
Invalid
Enter a phone number: 3X1-447-5820
Invalid
Enter a phone number: 3014475820
Invalid
```

13. Let `L` be a list of strings. Write list comprehensions that create new lists from `L` for each of the following.

    (a) A list that consists of the strings of `s` with their first characters removed

    (b) A list of the lengths of the strings of `s`

    (c) A list that consists of only those strings of `s` that are at least three characters long

14. Use a list comprehension to produce a list that consists of all palindromic numbers between 100 and 1000.

15. Use a list comprehension to create the list below, which consists of ones separated by increasingly many zeroes. The last two ones in the list should be separated by ten zeroes.

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

16. Let `L=[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47]`. Use a list comprehension to produce a list of the gaps between consecutive entries in `L`. Then find the maximum gap size and the percentage of gaps that have size 2.

17. Write a program that finds the average of all of the entries in a 4×4 list of integers.

18. Write a program that creates a 10×10 list of random integers between 1 and 100. Then do the following:

    (a) Print the list.

    (b) Find the largest value in the third row.

    (c) Find the smallest value in the sixth column.

19. Write a program that creates and prints an 8 × 8 list whose entries alternate between 1 and 2 in a checkerboard pattern, starting with 1 in the upper left corner.

20. Write a program that checks to see if a 4 × 4 list is a magic square. In a magic square, every row, column, and the two diagonals add up to the same value.

21. Write a program that asks the user to enter a length. The program should ask them what unit the length is in and what unit they would like to convert it to. The possible units are inches, yards, miles, millimeters, centimeters, meters, and kilometers. While this can be done with 25 if statements, it is shorter and easier to add on to if you use a two-dimensional list of conversions, so please use lists for this problem.

22. The following is useful as part of a program to play *Battleship*. Suppose you have a 5 × 5 list that consists of zeroes and ones. Ask the user to enter a row and a column. If the entry in the list at that row and column is a one, the program should print `Hit` and otherwise it should print `Miss`.

23. This exercise is useful in creating a *Memory* game. Randomly generate a 6 × 6 list of assorted characters such that there are exactly two of each character. An example is shown below.

```
@ 5 # A A !
5 0 b @ $ z
$ N x ! N z
0 - + # b :
- : + c c x
```

24. The following is useful in implementing computer players in a number of different games. Write a program that creates a 5 × 5 list consisting of zeroes and ones. Your program should then pick a random location in the list that contains a zero and change it to a one. If all the entries are one, the program should say so. [Hint: one way to do this is to create a new list whose items are the coordinates of all the ones in the list and use the `choice` method to randomly select one. Use a two-element list to represent a set of coordinates.]

25. Here is an old puzzle question you can solve with a computer program. There is only one five-digit number n that is such that every one of the following ten numbers shares exactly one digit in common in the same position as n. Find n.

$$01265, 12171, 23257, 34548, 45970, 56236, 67324, 78084, 89872, 99414$$

26. We usually refer to the entries of a two-dimensional list by their row and column, like below on the left. Another way is shown below on the right.

```
(0,0) (0,1) (0,2)     0  1  2
(1,0) (1,1) (1,2)     3  4  5
(2,0) (2,1) (2,2)     6  7  8
```

    (a) Write some code that translates from the left representation to the right one. The `//` and `%` operators will be useful. Be sure your code works for arrays of any size.

    (b) Write some code that translates from the right representation to the left one.