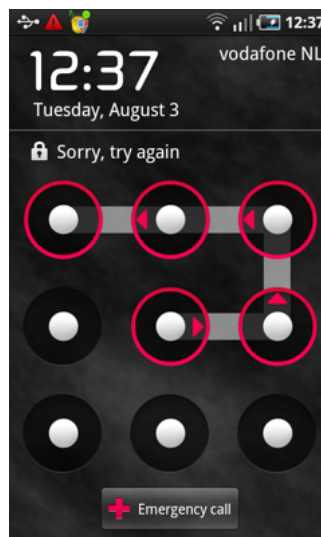




5. Week 5: Recursion and Permutations

5.1 Draw to Unlock

Rather than remembering passwords or passcodes, many mobile devices now allow the user to draw a pattern on the screen to unlock them.



Here we will explore how many unique patterns are available when drawing such patterns to connect “dots”, such as shown in the figure. We assume that people put their finger on one “dot” and then only ever move one position left, right, up or down (but never diagonally) at a time. **You are not allowed to return to a “dot” once it has been visited once.** If we number the first position in our path as 1, the second as 2 and so on, then beginning in the **top left-hand corner**, some of the possible **patterns of 9 moves** are :

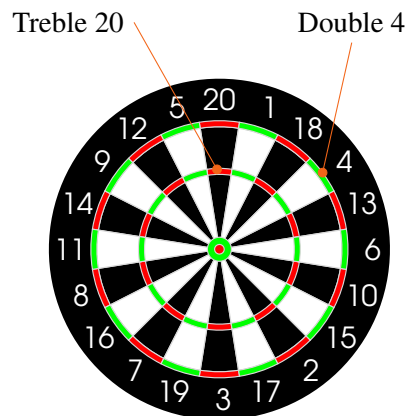
1 2 3	1 2 3	1 2 3
6 5 4	8 9 4	8 7 4
7 8 9	7 6 5	9 6 5

Exercise 5.1 Write a program that computes and outputs all the valid paths. Use **recursion** to achieve this.

- How many different patterns of **length 9** are available on a **3×3 grid**, if the user begins in the **top left corner** ?
- How many different patterns of **length 9** are available on a **3×3 grid**, if the user begins in the **middle left** ?
- How many different patterns of **length 7** are available on a **3×3 grid**, if the user begins in the **top left corner** ?
- How many different patterns of length 25 are available on a 5×5 grid, if the user begins in the top left corner ?

■

5.2 The Devil's Dartboard



In the traditional ‘pub’ game, darts, there are 62 different possible scores : **single 1 - 20** (the white and black areas), **double 1 - 20** (the outer red and green segments) (i.e. 2, 4, 6, 8 ...), **treble 1 - 20** (i.e. 3, 6, 9, 12 ...) (the inner red or green segments), **25** (small green circle) and **50** (the small red inner circle).

It’s not obvious, if you were inventing darts from scratch, how best to lay out the numbers. The London board shown seems to have small numbers near high numbers, so that if you just miss the 20 for example, you’ll hit a small number instead.

Here we look at a measure for the ‘difficulty’ of a dartboard. One approach is to simply sum up the values of adjacent triples, squaring this number. So for the London board shown, this would be: $(20 + 1 + 18)^2 + (1 + 18 + 4)^2 + (18 + 4 + 13)^2 \dots (5 + 20 + 1)^2 = 20478$

For our purposes a **lower** number is better¹. For more details see :

www

<http://www.mathpages.com/home/kmath025.htm>

Exercise 5.2 Write a program that repeatedly chooses two positions on the board and swaps their numbers. If this leads to a lower cost, keep the board. If not, unswap them. Repeat this *greedy search* 5000000 times, and print out the best board found. Begin with the trivial monotonic sequence. The output may look something like :

Total = 19966 : 3 19 11 2 18 12 1 20 10 4 16 8

¹It’s beyond the scope here to explain why!

14 5 13 15 6 7 17 9

or

Total = 19910 : 3 18 10 5 16 9 8 14 11 4 19 6
7 20 2 13 15 1 17 12

The score of 19874 seems to be the lowest possible that may be obtained. ■

5.3 Word Ladders

In this game, made famous by the author Lewis Carroll, and investigated by many Computer Scientists including Donald Knuth, you find missing words to complete a sequence. For instance, you might be asked how go from “WILD” to “TAME” by only changing one character at a time:

```
W I L D
W I L E
T I L E
T A L E
T A M E
```

A useful concept here is that of the *edit distance*. Here, the edit distance is a count of the number of characters which are different between two words. For words of n characters, the edit distance will be in the range $0 \dots n$. For ‘aboard’ and ‘canape’ the edit distance is 5. An edit distance of zero means that the words are identical.

In a heavily constrained version of this game we make some simplifying assumptions:

- Words are always four letters long.
- We only seek ladders of five words in total.
- Only one letter is changed at a time.
- A letter is only changed from its initial state, to its target state. This is important, since if you decide to change the second letter then you will always know what it’s changing from, to what it’s changing to.

So, in the example above, it is enough to give the first word, the last word, and the position of the character which changed on each line. On line one, the fourth letter ‘D’ was changed to an ‘E’, on the next line the first character ‘W’ was changed to a ‘T’ and so on. The whole ladder can be defined by “WILD”, “TAME” and the sequence 4, 1, 2, 3.

```
W I L D
W I L E
T I L E
T A L E
T A M E
```

Since each letter changes exactly once, the order in which this happens is a *permutation* of the numbers 1, 2, 3, 4, which we have looked at in Lectures.

Exercise 5.3 For the constrained version of the game, given a file of valid four letter words, write a program which when given two words on the command line (`argv[1]` and `argv[2]`) outputs the correct solution, if available. Use an exhaustive search over all 24 permutations until one leads to no invalid words being required. Make sure your program works, with

amongst others, the following:

C O L D
C O R D
C A R D
W A R D
W A R M

P O K E
P O L E
P O L L
M O L L
M A L L

C U B E
C U B S
T U B S
T U N S
T O N S

