Super Holiday Bonus Problems:

Question#1

As it turns out, Santa Clause is a major contributor to carbon emissions. His sled actually runs on diesel fuel. He burns tons of fuel each Christmas but each year is different depending on how much mass he has in his sled.

To find the fuel required for each year if he:

1. takes the mass of his loaded sleigh in tons, **divides by three**.
2. **rounds down to nearest whole number**, and then
3. **subtracts 2**

For example:

- For a mass of 13 tons, divide by 3, *round down* to get 4, then subtract 2 to get 2 tons.
- For a mass of 17, divide by 3, *round down* you get 5, subtract 2, you get 3 tons.
- For a mass of 1969, the fuel required is 654.
- For a mass of 100756, the fuel required is 33583.

Create a program for Santa that can take in the mass of presents in tons and output the amount of fuel required in tons.

Example:

```
What is the amount of toys this year in tons Santa:  2345

Great! You will need this many tons of diesel:  779
```

Hint: use **floor division**! `//` in python.

Question#2

Note: for this question you will have to learn (or relearn) how the ***binary number system*** works and how to convert binary number to decimal numbers.

Santa's Christmas workshop computer is now able to scan letters sent to the north pole and output a production code in the following form:

A10010111**R**00000111**T**10101010**D**11111101

The code is simple: Each **letter** in the `string eg.'A'` represents a type of toy that needs to be manufactured and the following number is an 8-bit binary version of a number representing the amount of toys needed to produce.

One problem is the production code is a `string` where the letters and numbers are `characters` and don't have numerical values.

Your job is to create a dictionary, or list of lists, or list of tuples where:

1. the **first** element is the **letter** (type of toy) and
2. the **second** element is the number of toys needed in **decimal** value.

The example above would have the output:

```
[['A', 151], ['R',7 ], ['T',170], ['D', 253]]
```

Question#3

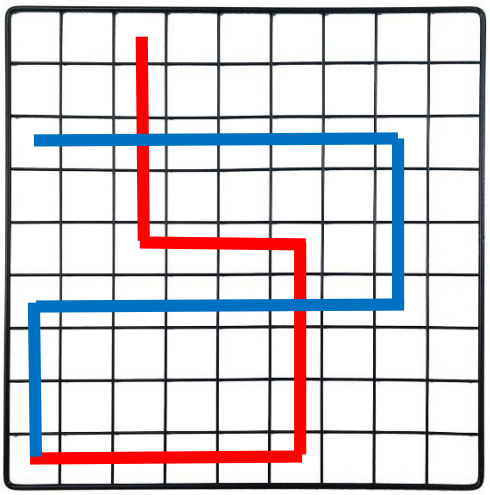
Santa's circuits:

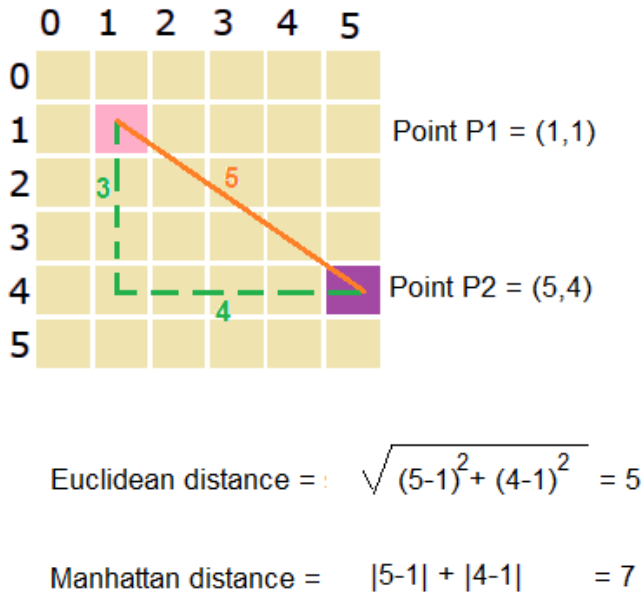
Santa hired Tesla Motor Corp to build him a new sleigh with auto pilot. Unfortunately one of the circuits always needs fixing. For the circuit in question, two wires (blue and red) always start from the same spot (0,0) as shown below.

The wires twist and turn, but the two wires occasionally cross paths. To fix the circuit, you need **to find the intersection point closest to the central port (0,0)**.

Because the wires are fixed on a grid, we will use the **Manhattan distance** for the "closest distance" measurement. See below.

While the wires do technically cross right at the central port (0,0) where they start, **this point does not count**, neither does a wire that **crosses with itself**.





Here are a two examples:

INPUT:

```
RED:  R5,U4,L3,U4
BLUE:  U3,R7,U3,L7
```

OUTPUT: `Closest wire cross is a Manhattan distance away of: 8`

INPUT:

```
RED:  R98,U47,R26,D63,R33,U87,L62,D20,R33,U53,R51
BLUE:  U98,R91,D20,R16,D67,R40,U7,R15,U6,R7
```

OUTPUT: `Closest wire cross is a Manhattan distance away of:  135`

Question#4

The password on the electronic door to Santa's workshop has been lost. The Elves had written the password on a sticky note, but someone threw it out.

However, they do remember a few key facts about the password:

- It is a six-digit number.
- Two adjacent digits are the same (like 22 in 122345).
- Going from left to right, the digits never decrease; they only ever increase or stay the same (like 111123 or 135679).

Examples:

- 111111 meets these criteria (double 11, never decreases).
- 223450 does not meet these criteria (decreasing pair of digits 50).
- 123789 does not meet these criteria (no double).

Several Elves come up to you to tell you they also seem to remember the password falling in between particular range of values.

Your job is:

Given a range of numbers (that are both 6 digits), determine how many different passwords are possible between this range.

Examples:

```
INPUT:
111123
111999
OUTPUT:  301 possible passwords (note: this value is not correct…just an example)

INPUT:
778999
888888
OUTPUT:  301 possible passwords (note: this value is not correct…just an example)
```