**Problem**

In this exercise, you will develop a function named decode(message\_file). This function should read an encoded message from a .txt file and return its decoded version as a string.

Guidelines for implementation:

The task is to create a function named **decode** that reads an encoded message from a .txt file and returns its decoded version as a string. The encoding is done using a pyramid structure in the input file.

1. The input file contains lines with a number followed by a word.

3 love

6 computers

2 dogs

4 cats

1 I

5 you

1. In this file, each line contains a number followed by a word. The task is to decode a hidden message based on the arrangement of these numbers into a "pyramid" structure. The pyramid increases by one number per line, like so:

  1

 2 3

4 5 6

1. The key to decoding the message is to use the words corresponding to the numbers at the end of each pyramid line (in this example, 1, 3, and 6). You should ignore all the other words.
2. So for the example input file above, the message words are: **1: I 3: love 6: computers** and your function should return the string

"I love computers".

**Answer: My approach**

1. Access the file
2. Convert the file data into a dictionary with keys(integers) and value(Strings) pairs.
3. Separate the keys into an array named, keys
4. Find the largest number in the array or sort the array to find the largest and whether the data is in correct order.
5. Then try to find how many rows of pyramid we can build with the number(largest).

Here, after lot of analysis I ended up with the formula.



We want to find T*n* such that Tn≤300*Tn*​≤300. We can set up the inequality.

<= 300

n = 24

The roots of this quadratic are n=24 and n = −25. Since we are looking for a positive value for n. Therefore n = 24

1. 

Also we can use this formula to find the ending number of the row.

1. But all this are mathematical, but we need to code it.

def decode(message\_file):

    # Initialize an empty dictionary to store the key-value pairs

    data\_dict = {}

    # Read the data from the file and populate the dictionary

    with open(message\_file, 'r') as file:

        for line in file:

            # Split each line into a list of strings

            line = line.strip().split()

            # Convert the first element of the list to an integer (key)

            key = int(line[0])

            # Take the second element of the list as the value

            value = line[1]

            # Add the key-value pair to the dictionary

            data\_dict[key] = value

    # Find the largest key in the dictionary

    largest\_key = max(data\_dict)

    # Initialize variables for loop control and tracking

    n = largest\_key

    rows = 1

    current\_number = 1

    last\_numbers = []

    # Generate a list of numbers representing the last number in each row

    while current\_number <= n:

        for \_ in range(rows):

            if current\_number <= n:

                current\_number += 1

            else:

                break

        last\_numbers.append(current\_number - 1)

        rows += 1

    # Example array of keys

    keys\_to\_access = last\_numbers

    # Access the values corresponding to the keys in the array

    values = [data\_dict[key] for key in keys\_to\_access]

    # Initialize a list to store the decoded words

    decoded\_words = []

    # Create a list of words based on the accessed values

    for key, value in zip(keys\_to\_access, values):

        decoded\_words.append(value)

    # Return the decoded message as a string

    return ' '.join(decoded\_words)

# Example usage:

decoded\_message = decode("coding\_qual\_input.txt")

print(decoded\_message.lower().strip())