Problem Statement - A1

Secret Message

You are trying to pass a sequence of numbers to your friend, but you do not want anyone else to see the contents of the message. What you need is a way to both encrypt and decrypt the message. One of the fastest ways to accomplish this is using a shift cipher. This encryption method takes a key (number) and will shift each digit in your message that many spots. Because the range of each digit is 0-9, the cipher should loop around if shifting the digit by the key causes it to leave this range. For example, the digit 4 with key 3 would be encrypted as 7; the digit 8 with key 5 will be encrypted as 3. Decryption works just like encryption but in reverse. So, the digit 7 with key 3 would be decrypted as 4; the digit 3 with key 6 would be decrypted as 7.

Write a program that takes in an encryption key, the number of digits to encrypt, and the value for each digit. The program should then print the encrypted sequence of digits on one line. Next, the program should ask for a decryption key and without any more input print the decrypted sequence of digits on one line. Note that the decrypted sequence comes from the just encrypted digits and not the original digits. You can assume that the value of each digit will be 0-9, and you can assume that any key given will be positive. You cannot assume that the keys will be less than 10.

Example 1	Example 2
Input: Enter encryption key: 4 **Input:** Number of digits: 3 **Input:** Enter digit 1: 8 **Input:** Enter digit 2: 2 **Input:** Enter digit 3: 6 **Output:** 260 **Input**: Enter decryption key: 6 **Output**: 604	**Input:** Enter encryption key: 67 **Input:** Number of digits: 4 **Input:** Enter digit 1: 1 **Input:** Enter digit 2: 7 **Input:** Enter digit 3: 9 **Input:** Enter digit 4: 3 **Output:** 8460 **Input**: Enter decryption key: 32 **Output**: 6248

Solution - A1

Secret Message

```
using System;

namespace ShiftCipherAdvanced
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] numbers;

            Console.Write("Enter encryption key: ");
            int key = Convert.ToInt32(Console.ReadLine());
            Console.Write("Number of digits: ");
            int n = Convert.ToInt32(Console.ReadLine());

            numbers = new int[n];

            for(int i = 0; i < n; i++)
            {
                Console.Write($"Enter digit {i}: ");
                numbers[i] = Convert.ToInt32(Console.ReadLine());
            }
            Console.Write("Encrypted: ");
            for (int i = 0; i < n; i++)
            {
                numbers[i] = (numbers[i] + key);
                while (numbers[i] >= 10) numbers[i] -= 10;
                Console.Write(numbers[i]);
            }

            Console.Write("\nEnter decryption key: ");
            key = Convert.ToInt32(Console.ReadLine());

            Console.Write("Decrypted: ");
            for (int i = 0; i < n; i++)
            {
                numbers[i] = (numbers[i] - key);
                while (numbers[i] < 0) numbers[i] += 10;
                Console.Write(numbers[i]);
            }
            Console.WriteLine();
            Console.ReadKey();
        }
    }
}
```

2 - Advanced - Backyard Overhaul

Problem Statement

This summer, the Griswald family would like to host the bi-weekly neighborhood barbeque. In order to prepare, they want to install a brand new square, two-toned patio in their back yard, but they aren't sure what will look best. They tell the construction company that whatever design they pick will be fine as long as it adheres to the following rules:

- Within any row or column there must be the same number of red bricks and tan bricks
- No row or column can have three consecutive bricks of the same color

Given an n by n grid of colored bricks, output whether or not the grid conforms to the Griswald family's rules.

Input

Your program should take the following input:

- An integer n, which represents the number of rows and columns in the grid, where $2 < n < 9$
- n strings of length n representing the colors of each brick (**R** for red and **T** for tan)

Output

Your program should output "True" or "False" indicating that the given grid of bricks conforms or does not conform to the specified rules.

Example

Input	Output
4 TRRT TRTR RTTR RTRT	True
4 RTTR RTTR TRRT TRTT	False

a2-backyard.py

```python
n = int(input("Enter the size of the grid: "))

grid = []
red = 0
tan = 0
consecutive = 0
last_color = ''

for r in range(1, n + 1):
    grid.append(input(f'Enter row {r}: '))


def check_brick(brick):
    global red, tan, consecutive, last_color
    if brick == 'R':
        red += 1
    else:
        tan += 1
    if last_color == brick:
        consecutive += 1
    else:
        consecutive = 1
        last_color = brick


correct = True
for row in grid:
    red = 0
    tan = 0
    last_color = ''
    consecutive = 1
    for c in row:
        check_brick(c)
        if consecutive >= 3:
            correct = False
    if red != tan:
        correct = False

if correct:
    for col in range(n):
        red = 0
        tan = 0
        last_color = ''
        consecutive = 1
        for row in grid:
            check_brick(row[col])
            if consecutive >= 3:
                correct = False

        if red != tan:
            correct = False

print(correct)

```

3 Advanced — Polynomials
Problem Statement

Every degree-n polynomial has exactly n roots, though some may not be real, and some may be duplicates. Given n roots $r_1, \ldots, r_n$, and a value a, a polynomial having exactly this set of roots may be computed as follows:

$$a(x - r_1) \cdots (x - r_n).$$

For example, if $n = 2$, $r_1 = 1$, $r_2 = 2$, and $a = 1$, the polynomial would be $x^2 - 3x + 2$.

Write a program that takes as input a positive integer n and n real roots (duplicates are allowed) and produces as output the coefficients of the polynomial having these roots, assuming $a = 1$. You must list the coefficients in order from the x^n term to the constant term. You may assume that $n \leq 10$.

Example 1:

```
Enter n: 2
Enter root1: 1
Enter root2: 2

1 -3 2
```

Example 2:

```
Enter n: 5
Enter root1: -2.5
Enter root2: 1.5
Enter root3: -1
Enter root4: 3
Enter root5: -1

1 0 -9.75 -4.25 15.75 11.25
```

```
1  // 3 Advanced - Polynomials
2
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace PolynomialsAdvanced
10 {
11     class Program
12     {
13         static void Main(string[] args)
14         {
15             Console.Write("Enter n: ");
16             int n = Convert.ToInt32(Console.ReadLine());
17             float[] p = new float[1];
18             p[0] = 1;
19             for (int i = 1; i <= n; i++)
20             {
21                 float[] newP = new float[i + 1];
22                 Console.Write("Enter root" + i + ": ");
23                 float r = Convert.ToSingle(Console.ReadLine());
24                 newP[0] = p[0] * -r;
25                 newP[i] = p[i - 1];
26                 for (int j = 1; j < i; j++)
27                 {
28                     newP[j] = p[j - 1] - p[j] * r;
29                 }
30                 p = newP;
31             }
32             Console.WriteLine();
33             for (int i = n; i >= 0; i--)
34             {
35                 Console.Write(p[i] + " ");
36             }
37             Console.ReadLine();
38         }
39     }
40 }
41
```

Advanced Problem Statement A4

Traffic Signals

To help the flow of traffic, some traffic signals are synchronized so there is a “sweet speed” that allows traffic to go through an unlimited series of such traffic signals without changing speed. This means that the sweet speed must exactly match the synchronization. If vehicle is slightly faster than the sweet speed, it will eventually arrive at a traffic signal before the light has turned green. If the speed is too slow it will eventually miss the green light. It is usually planned that the sweet speed is well within the speed limits.

Consider a situation where there are 20 traffic signals X miles apart on a straight section of highway. Each traffic signal has a red signal for 20 seconds followed by green signal for G seconds. (Ignore the yellow signal. Assume it is part of the end of the green signal). The each traffic signal turns green D seconds after the previous traffic signal turns green. The speed limit on the road is 70 miles per hour and the minimum speed is 45 miles per hour. Assume that sweet speed does not involve multiple sequences of red/green during the vehicle moving between traffic signals.

Suppose a car approaches the first traffic signal in the midpoint of the green. What is the legal sweet speed that the car must maintain to go through the all the traffic signals on a green signal? If the car cannot go through all the traffic signals, what is the maximum or minimum speed in which the car goes through at least N of the traffic signals. (Hint) If the desired sweet speed is above 70, then calculate the minimum speed necessary to go through the Nth traffic signal. Output “Not Possible” if that speed is still greater than the legal speed. Do similar for the minimum limit of 45.

Assume that the car can instantly achieve the appropriate speed and accurately maintain it for the 20*X miles. Distance X is given in miles. Delay D and Green light time G is given in seconds. Output is in miles per hour. All times and distances are real numbers. Only N is integer.

Ex 1:	X = 1	Ex 2:	X = 1.2	Ex 3:	X = 1.3	Ex 4:	X = 0.7
	D = 60		D = 60		D = 60		D = 60
	G = 30		G = 30		G = 30		G = 60
	N = 4		N = 4		N = 4		N = 5
Out:	Sweet 60		Min is 66.4615		not possible		Max is 48

A4
Traffic Signals
Source Code

```cpp
#include <iostream>

int main()
{
    char Q; Q = 'N';
    while (Q == 'N')
    {
        double N,S,X,G,D;
        std::cout << "Hello World!\n";
        std::cout << "\nenter distance X ";
        std::cin >> X;
        std::cout << "\nenter delay D ";
        std::cin >> D;
        std::cout << "\nenter green duration G ";
        std::cin >> G;
        std::cout << "\nenter number of traffic signals N ";
        std::cin >> N;
        S = (X * 60 * 60) / (D);
        std::cout << "\nS = " << S;
        if (30 < S && S < 70) { std::cout << "\nSweet speed is "; std::cout << S; }
        else if (S < 30) {
            S = ((N - 1) * X * 3600) / (((N - 1) * D )-(0.5 * G));
            if (S < 30) { std::cout << "\n" << N << " not possible"; }
            else { std::cout << "\nMax Speed = " << S; }
        }
        else if (S > 70) {
            S = ((N - 1) * X * 3600) /( ((N - 1) * D)+(0.5 * G));
            if (S > 70) { std::cout << "\n" << N << " not possible"; }
            else { std::cout << "\nMin Speed = " << S;
            }
        }
        std::cout << "Quit?";
        std::cin >> Q;
    }
```

5 - Advanced - Dr. Indigo's Zoo

Problem Statement

In their free time, when they're not busy saving the world, Dr. Indigo runs a zoo. Every morning they get up and makes sure that none of the animals have escaped. They have a huge list of all the animals and checks each animal off as they see it but thinks this is really inefficient. They only care about what animal they are, since all similar animals share a cage. So, if there is a white tiger and a Siberian tiger, Dr. Indigo only wants to know that they have 2 tigers.

Given an integer n, and n lines describing animals, output each type of animal Dr. Indigo has in the zoo, followed by their count.

Input

Your program should take the following input:

- An integer n, which represents the total number of animals, where $0 < n < 10$
- n strings, each describing an animal. The last word in the string will represent the animal type. You may assume that all input will be lower case letters. Each word in the animal description will be separated by a single space. Each animal description will contain 1 or more words.

Output

Your program should output each type of animal along with their count. Output does not have to be in any particular order.

Example

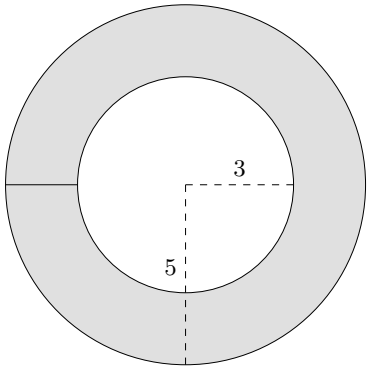
Input	Output
7 african elephant white tiger indian elephant siberian tiger tiger panda bear blue russian penguin	elephant: 2 tiger: 3 bear: 1 penguin: 1

a5-zoo.py

```python
# read number of animals
x = int(input("Enter number of animals: "))

# dictionary to store animals
# this can be solved utilizing lists/arrays, or more efficiently with a dictionary
animals = {}
for i in range(x):
    # get last word of line in lower case
    line = input(f'Enter animal {i + 1}: ').split()[-1].lower()

    # add to dictionary or increment
    if line in animals:
        animals[line] += 1
    else:
        animals[line] = 1

# print list
for key in animals:
    print(f"{key}: {animals[key]}")

```

6 Advanced — Space Race
Problem Statement

A space ship captain is preparing for a race on the course shown to the right. The course lies between two concentric circles, the inner one having a radius of 3, and the outer one having a radius of 5. The center of these circles lies at coordinates $(0, 0)$. The start/finish line is the line segment from $(-5, 0)$ to $(-3, 0)$. The ship starts at $(-4, 0)$ and travels in a counter-clockwise direction.



The ship begins at rest. At the beginning of the race, and every second thereafter, the captain can apply an acceleration in any direction. This acceleration lasts for a duration of one second, until the next acceleration is applied. The ship must stay between the two circles, or it will be disqualified.

Write an interactive program that takes the accelerations to apply at each second, and after each acceleration is applied for one second, outputs the ship's current position. Each acceleration consists of two floating-point values, each at least -1 and at most 1. The first value gives the horizontal component of the acceleration, with positive values acceleration toward the right, and the second value gives the vertical component, with positive values accelerating downward. If at any point the ship leaves the bounds of the race course, display a message, "Out of bounds", followed by the point at which it was detected to be out of bounds, and stop the simulation. Note that the ship is within the bounds if $3^2 \leq x^2 + y^2 \leq 5^2$. Because the ship may leave the course and reenter it before the current second elapses, check the position of the ship after each 0.01 seconds. Use the following formulas (separately in each dimension) to update the position and speed, where t represents the length of the time interval, d represents the net change in position over the time interval, v represents the speed at the beginning of the time interval, v' represents the speed at the end of the time interval, and a represents the acceleration applied:

$$d = vt + at^2/2$$
$$v' = v + at$$

Stop the simulation after 5 seconds have elapsed if the ship is still within the course bounds.

In the examples that follow, output is shown in italics. Your output should match the output shown to three significant digits.

Example 1:

```
Enter horizontal acceleration: 1
Enter vertical acceleration: 1
-3.5, 0.4999997
Enter horizontal acceleration: 1
Enter vertical acceleration: 1
Out of bounds: -2.703951, 1.296049
```

Example 2:

```
Enter horizontal acceleration: 1
Enter vertical acceleration: 1
-3.5, 0.4999997
Enter horizontal acceleration: 0
Enter vertical acceleration: 0.5
-2.500001, 1.749999
Enter horizontal acceleration: 0
Enter vertical acceleration: 0
-1.500002, 3.250007
Enter horizontal acceleration: 0.5
Enter vertical acceleration: -1
-0.2500031, 4.250006
Enter horizontal acceleration: 0
Enter vertical acceleration: -1
1.249996, 4.250006
```

```
1  // 6 Advanced - Space Race
2
3  using System;
4  using System.Collections.Generic;
5  using System.Linq;
6  using System.Text;
7  using System.Threading.Tasks;
8
9  namespace SpaceRaceAdvanced
10 {
11     class Program
12     {
13         static void Main(string[] args)
14         {
15             float x = -4;
16             float y = 0;
17             float hSpeed = 0;
18             float vSpeed = 0;
19             for (int i = 0; i < 5; i++)
20             {
21                 Console.Write("Enter horizontal acceleration: ");
22                 float hAccel = Convert.ToSingle(Console.ReadLine());
23                 Console.Write("Enter vertical acceleration: ");
24                 float vAccel = Convert.ToSingle(Console.ReadLine());
25                 for (int j = 0; j < 100; j++)
26                 {
27                     x += hSpeed * 0.01f + hAccel * 0.00005f;
28                     y += vSpeed * 0.01f + vAccel * 0.00005f;
29                     float dSq = x * x + y * y;
30                     if (dSq < 9 || dSq > 25)
31                     {
32                         Console.WriteLine("Out of bounds: " + x + ", " + y);
33                         Console.ReadLine();
34                         return;
35                     }
36                     hSpeed += hAccel * 0.01f;
37                     vSpeed += vAccel * 0.01f;
38                 }
39                 Console.WriteLine(x + ", " + y);
40             }
41             Console.ReadLine();
42         }
43     }
44 }
45
```