B1 Problem Statement

Willie's Push-ups

After every football score, Willie the Wildcat does one push-up for every point scored so far in the current game.

Write a program that accepts four positive integer scores and prints the total number of push-ups that Willie has to do. Assume that the score was zero before the first score.

```
Example 1: Input:     Score 1: 3
                      Score 2: 7
                      Score 3: 7
                      Score 4: 3
           Output:    Total push-up: 50

Example 2: Input:     Score 1: 3
                      Score 2: 7
                      Score 3: 3
                      Score 4: 3
           Output:    Total push-up: 42

Example 3: Input:     Score 1: 3
                      Score 2: 3
                      Score 3: 7
                      Score 4: 7
           Output:    Total push-up: 42
```

B2 Problem Statement

Muddy Paws

There have been a few rain storms the last couple of days, but the puppies still had to go outside to play. Coming back inside, the puppies left muddy paw prints all over the house. You are tasked with trying to clean as many rooms of the house as you can, but you only have so much cleaner. Write a program to determine how many of the three rooms in the house you can clean if you have 10 gallons of cleaner. Each gallon of cleaner can clean a maximum of 500 square feet. Your program should first take input for the square footage (given as a whole number) of each room. The inputs are given in ascending order of size. Inputs are given one per line. Finally, your program should output the number of rooms you can clean.

Example

```
Enter the square footage of room 1: 500
Enter the square footage of room 2: 3000
Enter the square footage of room 3: 4000
2 room(s) were cleaned.
```

```
Enter the square footage of room 1: 100
Enter the square footage of room 2: 200
Enter the square footage of room 3: 300
3 room(s) were cleaned.
```

3 Beginning — Minimum Distance

Write a program that takes as input three points, a, b, and c, in the 2-dimensional plane and determines whether a is closer to b or to c. The distance between two points (x_1, y_1) and (x_2, y_2) is given by

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

(Note that you don't have to compute the square root to solve the problem, as the square of the distance is minimized whenever the distance is minimized.) All inputs will be floating-point numbers.

Example:

```
Enter x-coordinate of a: 0
Enter y-coordinate of a: 0
Enter x-coordinate of b: 0.5
Enter y-coordinate of b: -0.5
Enter x-coordinate of c: -0.5
Enter y-coordinate of c: 0

a is closer to c.
```

B4 Problem Statement

Adding Fractions

To add fractions, you must convert the fractions to a common denominator, add the numerators and then convert the result to the lowest denominator.

e.g. 1/4 + 1/2 = 2/8 + 4/8 = 6/8 = 3/4 or a/b + c/d = ad/bd + cb/bd = (ad + cb) / bd

The fraction (ad + cb) / bd then needs to be reduced by dividing both the numerator and denominator by the largest integer that evenly divides both.

Write a program that accepts the first numerator and denominator and the second numerator and denominator, then displays the sum expressed in whole numbers plus a fractional part expressed in the lowest denominator. A fraction with a zero numerator will not be displayed.

All inputs and outputs must be integers. The fractions must be represented either by an integer numerator followed by a "/" and an integer denominator or by the numerator and denominator printed separately and labelled as numerator and denominator respectively, e.g. "numerator is 3 ; denominator is 4".

All inputs will be positive integers, i.e. greater than zero.

Example 1:

Input:	Num1: 1
	Den1: 4
	Num2: 1
	Den2: 2
Output:	Fraction: 3/4

Example 2:

Input:	Num1: 3
	Den1: 6
	Num2: 1
	Den2: 2
Output:	Fraction: 1

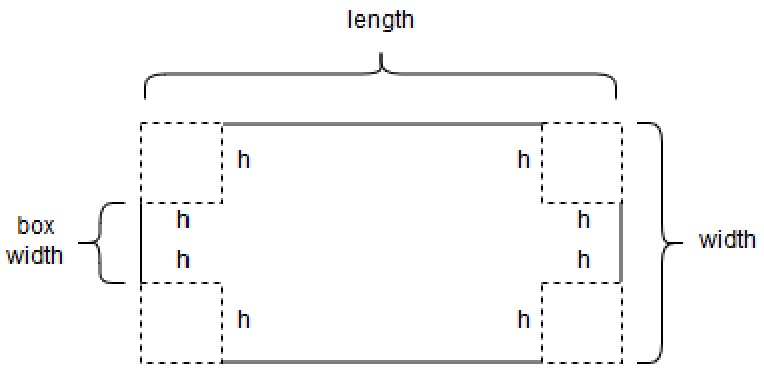
Example 3:

Input:	Num1: 7
	Den1: 2
	Num2: 7
	Den2: 3
Output:	Fraction: 5 5/6

B5 Problem Statement

Packing Up Decorations

It is time to start packing up all of the holiday decorations, but we lost all of the boxes! All we have are some flat pieces of cardboard. We can use the cardboard to make a box (with no lid.) We already know how wide the box should be. Write a program that takes in the width and length of a rectangular piece of cardboard, as well as the width of the desired box. Use this information to calculate the height, length, and volume of the box. The box should hold as many decorations as possible. The box should be made by cutting a square out of each corner of the cardboard like in the figure below, where h is the height of the box.



Input for this program will be given one per line. The length of the cardboard is given first, then the width, followed by the width of the box. All values will be given in inches as floating point values. You may assume that the dimensions entered for the cardboard represent a valid rectangle where the width represents the short side and is at least at least 3 inches. However, you may not assume that the given box width creates a valid box. Additionally, the box should be at least one inch high. If the box created from the input is invalid, the program should print a message to indicate so, otherwise, your program should should output the dimensions (length, width, height) and volume of the box. All numerical output should be rounded to two decimal places.

Example

```
Enter length of the cardboard: 10.0
Enter width of the cardboard: 5.0
Enter width of the box: 3.0

length = 8.00
width = 3.00
height = 1.00
volume = 24.00
```

```
Enter length of the cardboard: 15.5
Enter width of the cardboard: 9.3
Enter width of the box: 4.5

length = 10.70
width = 4.50
height = 2.40
volume = 115.56
```

```
Enter length of the cardboard: 10
Enter width of the cardboard: 9
Enter width of the box: 8
Those dimensions do not create a valid box.
```

6 Beginning — Next Card

This problem is based on a 1-player game consisting of n cards containing the values 0 through $n - 1$. The cards are shuffled and placed face down. The player then reveals one card at a time, and for each card, guesses whether the next card's value will be higher or lower than the last. If she guesses all cards correctly, she wins; otherwise, she loses.

Write a program that takes as input the number of cards and the first two cards of the game, and produces as output the probability that the next card is lower than the second card and the probability that the next card is higher than the second card. The probability that a randomly-chosen card is in a given subset of the remaining cards is given by dividing the number of cards in the subset by the number of cards remaining. You may assume that the number of cards will be at least 3, and that the first two cards are distinct and in the specified range. All inputs will be integers.

Example:

```
Enter number of cards: 6
Enter first card: 2
Enter second card: 4

Probability the next card is lower: 0.75
Probability the next card is higher: 0.25
```

B1 Solution

Willie’s Push-ups

```
int main()
{

       int S1, S2, S3, S4, CP;
       char q; q = 'a';
       CP = 0;
       q = 'a';
              while (q != 'N')
              {
                     std::cout << "\nEnter first score: "; std::cin >> S1;
                     std::cout << "\nEnter second score: "; std::cin >> S2;
                     std::cout << "\nEnter third score: "; std::cin >> S3;
                     std::cout << "\nEnter fourth score: "; std::cin >> S4;

                     CP = 4 * S1 + 3 * S2 + 2 * S3 + S4;

                     std::cout << "\nWillie did " << CP << " push-ups";

              std::cout << "\nAnother Game?: ";  std::cin >> q;
       }

       return 0;
}
```

```
gpsqft = 500
gallons = 10
cleaned = 0

rooms = []
i = 1
while i <= 3:
    rooms.append(int(input("Enter the square footage of room {}: ".format(i))))
    i += 1

for room in rooms:
    used = room / gpsqft
    if used <= gallons:
        cleaned += 1
        gallons -= used

print("{} room(s) were cleaned.".format(cleaned))
```

B3 Solution

```
/* 3 Beginning - Minimum Distance
 * Author: Rod Howell
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MinimumDistance.Beginning
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter x-coordinate of a: ");
            double aX = Convert.ToDouble(Console.ReadLine());
            Console.Write("Enter y-coordinate of a: ");
            double aY = Convert.ToDouble(Console.ReadLine());
            Console.Write("Enter x-coordinate of b: ");
            double bX = Convert.ToDouble(Console.ReadLine());
            Console.Write("Enter y-coordinate of b: ");
            double bY = Convert.ToDouble(Console.ReadLine());
            Console.Write("Enter x-coordinate of c: ");
            double cX = Convert.ToDouble(Console.ReadLine());
            Console.Write("Enter y-coordinate of c: ");
            double cY = Convert.ToDouble(Console.ReadLine());
            double xDistToB = aX - bX;
            double yDistToB = aY - bY;
            double xDistToC = aX - cX;
            double yDistToC = aY - cY;
            Console.WriteLine();
            if (xDistToB * xDistToB + yDistToB * yDistToB <
                xDistToC * xDistToC + yDistToC * yDistToC)
            {
                Console.WriteLine("a is closer to b.");
            }
            else
            {
                Console.Write("a is closer to c.");
            }
            Console.ReadLine();
        }
    }
}

```

B4 Solution

Adding Fractions

```
int main()
{
	int N1, N2, D1, D2, FN, FD;
	int W;
	int i;
	char q; q = 'a';

	q = 'a';
	while (q != 'N')
	{
		W = 0;
		std::cout << "\nEnter first numerator: "; std::cin >> N1;
		std::cout << "\nEnter first denominator: "; std::cin >> D1;
		std::cout << "\nEnter second numerator: "; std::cin >> N2;
		std::cout << "\nEnter second denominator: "; std::cin >> D2;

		FN = N1*D2 + N2*D1;
		FD = D1*D2;

		while (FN >= FD) {
			W = W + 1; FN = FN - FD;
		}
		for (i = FN - 1; i > 1; i = i - 1) {
			if ((((FN / i)* i) == FN) && (((FD / i)*i) == FD))
				{FN = FN / i; FD = FD / i;
				}
		}
		if (FN > 0)
		{
			std::cout << "\nFinal Fraction is " << W << " plus "
				      << FN << "/" << FD;
		}
		else
		{
			std::cout << "\nFinal Fraction is " << W;


		}
		std::cout << "\nAnother Game?: ";  std::cin >> q;
	}

	return 0;

}
```

B5 Solution

```python
length = float(input("Enter length of the cardboard: "))
width = float(input("Enter width of the cardboard: "))
box_width = float(input("Enter width of the box: "))
height = (width-box_width)/2
length = length-height*2
volume = box_width * length * height
if box_width <= 0 or box_width >= width or height < 1:
    print("Those dimensions do not create a valid box.")
else:
    print("\nlength = {:.2f}\nwidth = {:.2f}\nheight = {:.2f}\nvolume = 
    {:.2f}".format(length, box_width, height, volume))
```

B6 Solution

```
/* 6 Beginning - Next Card
 * Author: Rod Howell
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace NextCard.Beginner
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter number of cards: ");
            int n = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter first card: ");
            int card1 = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter second card: ");
            int card2 = Convert.ToInt32(Console.ReadLine());
            int nLower = card2;
            if (card1 < card2)
            {
                nLower--;
            }
            Console.WriteLine();
            Console.WriteLine("Probability the next card is lower: " +
                (float)nLower / (n - 2));
            Console.WriteLine("Probability the next card is higher: " +
                (float)(n - nLower - 2) / (n - 2));
            Console.ReadLine();
        }
    }
}

```