# A1 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.  On the scoreboard at the stadium, there is displayed the total number of push-ups that Willie has done since the beginning of the season.

Write a program that accepts the current total number of push-ups (from the previous game).  The program than accepts the first score of the game and then prints the number of push-ups that Willie has to do after this score and the current game push-up total. Keep prompting until a zero indicates the end of the game.  Then output the new total for the season.

```
Example 1: Input:    Previous total: 90
                     New score: 7
          Output:    Current push-ups: 7
                     Game push-up total:   7
           Input:    New score:  3
           Output:   Current push-ups: 10
                     Game push-up total: 17
           Input:    New score: 0
           Output:   New total: 107

 Example 2: Input:   Previous total: 105
                     New score: 3
          Output:    Current push-ups:         3
                     Game push-up total:   3
           Input:    New score:  3
           Output:   Current push-ups: 6
                     Game push-up total: 9
           Input:    New score:  6
           Output:   Current push-ups: 15
                     Game push-up total: 24
           Input:    New score: 0
           Output:   New total: 129
```

# A2 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 rooms 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 in input for how many rooms there are, followed by input for the square footage (given as a whole number) of each room. The rooms are not given in order of size. Inputs are given one per line. Finally, your program should output the number of rooms you can clean.

## Example

```
How many rooms are there? 3
Enter the square footage of room 1: 3000
Enter the square footage of room 2: 2000
Enter the square footage of room 3: 4000
2 room(s) were cleaned.
```

---

```
How many rooms are there? 2
Enter the square footage of room 1: 800
Enter the square footage of room 2: 1200
2 room(s) were cleaned.
```

## 3 Advanced — Minimum Distance

Write program that takes as input a number $n$ of points (an integer), followed by the $x$- and $y$-coordinates (floating point) of $n$ points, and produces as output the closest pair of these points. 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 need to compute the square root to solve this problem, as the square of the distance is minimized whenever the distance is minimized.) You may assume that $n$ will be at least 2 and at most 10.

**Example:**

```
Enter number of points: 4
Enter x0: -1.5
Enter y0: 3.7
Enter x1: 2.3
Enter y1: 5.6
Enter x2: 1.2
Enter y2: -10
Enter x3: 2.4
Enter y3: 5.1

(2.3, 5.6) and (2.4, 5.1) are closest.
```

# A4 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 a series of numerator and denominator pairs, adds each pair to the previous result and expresses the new sum in whole numbers plus a fractional part expressed in the lowest denominator. A fraction with a zero numerator will not be displayed. No output is required until after the second numerator and denominator.

**All inputs and outputs must be integers.** The fractions must be represented either 1) by an integer numerator followed by a "/" and an integer denominator, e.g. "3/4", or 2) 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:  2
                      Dem1:  4
                      Num:   3
                      Dem:   4
           Output:     1 1/4
           Input:     Num:   2
                      Dem:   9
           Output:     1  17/36
```

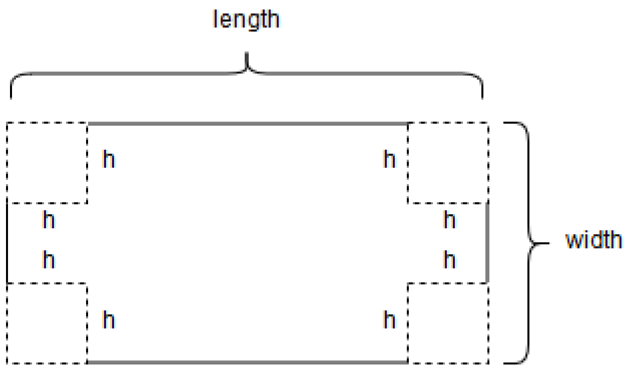
```
Example 2: Input:     Num1:  83
                      Dem1:  99
                      Num:   6
                      Dem:   33
           Output:     1 2/99
```

```
Example 3: Input:     Num1:  83
                      Dem1:  99
                      Num:   16
                      Dem:   99
           Output:     1
```

# A5 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). Write a program that takes in the width and length of a rectangular piece of cardboard and calculates the dimensions of a box. This box should be able to hold as many decorations as possible. Therefore, your program should maximize the volume of the box using only whole number heights. As seen in the figure below, the box can be made from the carboard by cutting out a square, $hxh$, from each corner where $h$ represents 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. All values will be given as whole inches. 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. Your program should should output the dimensions (length, width, height) and volume of the box you create.

## Example

Enter length of the cardboard: 8
Enter width of the cardboard: 3

length = 6
width = 1
height = 1
volume = 6

---

Enter length of the cardboard: 25
Enter width of the cardboard: 20

length = 17
width = 12
height = 4
volume = 816

## 6 Advanced — 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, then the value of each card in sequence. Following each card, it produces as output the best guess (i.e., lower or higher than the last card), along with the probability that this guess is correct. 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. If the probabilities of both guesses are the same, the program should guess "lower". If any guess is incorrect, the program should immediately terminate with the message, "I lose.". If it guesses all cards correctly, it should display the message, "I win!". You may assume that the number of cards will be at least 2, and that each guess is distinct and in the specified range. All inputs will be integers.

**Example 1:**

```
Enter number of cards: 10
Enter next card: 5
Lower (probability = 0.5555556)
Enter next card: 4
Lower (probability = 0.5)
Enter next card: 8
I lose.
```

**Example 2:**

```
Enter number of cards: 4
Enter next card: 1
Higher (probability = 0.6666667)
Enter next card: 3
Lower (probability = 1)
Enter next card: 0
Higher (probability = 1)
Enter next card: 2
I win!
```

A1 Solution

Willie’s Push-ups

```
int main()
{
	float NS, CP, CS, PTP;

	char q; q = 'a';
	char ng; ng = 'b';
	while (ng != 'N')
	{
		CP = 0; CS = 0;
		q = 'a';
		std::cout << "\nEnter prior total push-ups: "; std::cin >> PTP;

		while (q != 'N')
		{
			std::cout << "\nEnter new score: "; std::cin >> NS;
			if (NS == 0) {
				PTP = PTP + CP;
				std::cout << "\nEnd of Game - New total push-ups: "
					<< PTP << "\n";
				q = 'N'; //ng = 'N'; //return 0;
			}
			else
			{
			CS = CS + NS;
			CP = CP + CS;

			std::cout << "\nCurrent push-ups: " << CS
				<< " Game push-up total: "; std::cout << CP;
			}
		}

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

	return 0;}
```

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

n = int(input("How many rooms are there? "))
rooms = []
i = 1
while i <= n:
    rooms.append(int(input("Enter the square footage of room {}: ".format(i))))
    i += 1

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

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

```

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

namespace MinimumDistance.Advanced
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter number of points: ");
            int n = Convert.ToInt32(Console.ReadLine());
            double[] x = new double[n];
            double[] y = new double[n];
            for (int i = 0; i < n; i++)
            {
                Console.Write("Enter x" + i + ": ");
                x[i] = Convert.ToDouble(Console.ReadLine());
                Console.Write("Enter y" + i + ": ");
                y[i] = Convert.ToDouble(Console.ReadLine());
            }
            double min = Double.PositiveInfinity;
            int index1 = -1;
            int index2 = -1;
            for (int i = 0; i < n; i++)
            {
                for (int j = i + 1; j < n; j++)
                {
                    double diffX = x[i] - x[j];
                    double diffy = y[i] - y[j];
                    double distSq = diffX * diffX + diffy * diffy;
                    if (distSq < min)
                    {
                        index1 = i;
                        index2 = j;
                        min = distSq;
                    }
                }
            }
            Console.WriteLine();
            Console.Write("(" + x[index1] + ", " + y[index1] + ") and (" +
                x[index2] + ", " + y[index2] + ") are closest.");
            Console.ReadLine();
        }
    }
}

```

A4 Solution

Adding Fractions

```
int main()
{
	float RN, RF, R1, R2;
	int N1, N2, D1, D2, FN, FD;
	int W, i;
	char q; q = 'a';
	char n; n = 'n';

	while (n != 'N') {
		std::cout << "\nEnter first numerator: "; std::cin >> N1;
		std::cout << "\nEnter first denominator: "; std::cin >> D1;
		R1 = N1; R2 = D1;  RF = R1 / R2; std::cout << "\n floatF:" << RF;
		q = 'a';
		while (q != 'N')
		{
			W = 0;
			std::cout << "\nEnter next numerator: "; std::cin >> N2;
			std::cout << "\nEnter next denominator: "; std::cin >> D2;
			R1 = N2; R2 = D2;  RN = R1 / R2; std::cout << "\n floatN: " << RN;

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

			while (FN >= FD) {
				W = W + 1; FN = FN - FD;
			}
			for (i = FD - 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 << "\nCurrent Fraction is " << W << " plus "
					<< FN << "/" << FD;
			}
			else
			{
				std::cout << "\nCurrent Fraction is " << W ;
			}
			RF = RF + RN; std::cout << "\n floatF:" << RF;

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

		std::cout << "\nStart a new Fraction?: ";  std::cin >> n;

	}
	return 0;

}
```

A5 Solution

```python
length = int(input("Enter length of the cardboard: "))
width = int(input("Enter width of the cardboard: "))
volume = 0
h = 0
max_volume = -1
max_height = -1
small_side = min(width, length)
while h * 2 < width:
    h += 1
    volume = h * (length - 2 * h) * (width - 2 * h)
    if volume > max_volume:
        max_volume = volume
        max_height = h
print("\nlength = {:.0f}\nwidth = {:.0f}\nheight = {:.0f}\nvolume = 
{:.0f}".format(length - 2 * max_height, width - 2 * max_height, max_height, max_volume))
```

A6 Soltuion

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

namespace NextCard.Advanced
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter number of cards: ");
            int n = Convert.ToInt32(Console.ReadLine());
            bool[] used = new bool[n];
            int last = -1;
            bool lower = false;
            while (true)
            {
                Console.Write("Enter next card: ");
                int i = Convert.ToInt32(Console.ReadLine());
                used[i] = true;
                n--;
                if ((lower && i > last) || (!lower && i < last))
                {
                    Console.WriteLine("I lose.");
                    Console.ReadLine();
                    return;
                }
                else if (n == 0)
                {
                    Console.WriteLine("I win!");
                    Console.ReadLine();
                    return;
                }
                int less = 0;
                for (int j = 0; j < i; j++)
                {
                    if (!used[j])
                    {
                        less++;
                    }
                }
                if (less >= n - less)
                {
                    Console.WriteLine("Lower (probability = " +
                        (float)less / n + ")");
                    lower = true;
                }
                else
                {
                    Console.WriteLine("Higher (probability = " +
                        (float)(n - less) / n + ")");
                    lower = false;
                }
                last = i;
            }
        }
    }
}

```