1 Beginning — Cruising to K-State

Problem Statement

After a nice Thanksgiving break, you are ready to head back to K-State. Before you leave, you want to see how much time it will take to get to K-State from your house so you can leave at the perfect time. You know that Google Maps can easily calculate how much time it will take, but you also know that you might have to stop for gas costing you time. Thinking back to an article you read, you remember that you use more gas the faster you go, so with this idea, you have decided to determine how much time it will take with gas stops included to get back to Manhattan.

Write a program that accepts the distance(in miles) to Manhattan and the driving speed(in miles per hour). The program then displays how many gas stops you made and how much time it took to drive back to Manhattan. All values are allowed a maximum error of .01 (ie: for an answer of 1.78, 1.79 or 1.77 will suffice).

Important to know

- Assume that the car has a 15.0 gallon gas tank and you are always going at least 55.0 mph.

- Every time you refill, you lose 0.5 hours as you stop for gas, refill, get up to speed, etc.

- As speed increases, the amount of power required to overcome drag, friction, and rolling resistance increases. This results in a lower fuel efficiency as more fuel is used to cover less distance regardless of how fast the car is going. Thus, knowing that on average the best fuel efficiency of 24.7 miles per gallon (mpg) is achieved around 55.0 mph, use the equation *mpg = 39 - (0.26 * speed)* to approximate your mpg.

Example 1	Example 2
Input: Enter distance(mi) to Manhattan, KS: 170.1 **Input:** Enter your speed(mph): 75.0 **Output:** You'll have to stop for gas 0 time(s). **Output:** With that in mind, it will take you 2.27 hours to get back to Manhattan!	**Input:** Enter distance(mi) to Manhattan, KS: 512.7 **Input:** Enter your speed(mph): 55.5 **Output:** You'll have to stop for gas 1 time(s) **Output:** With that in mind, it will take you 9.74 hours to get back to Manhattan!

1 Beginning — Cruising to K-State

Solution

```
static void Main(string[] args)
{
    Console.Write("Enter distance(mi) to Manhattan, KS: ");
    double distance = Double.Parse(Console.ReadLine());

    Console.Write("Enter your speed(mph): ");
    double speed = Double.Parse(Console.ReadLine());
    double mpg = (-0.26 * speed) + 39;

    int gasTank = 15;

    int gasStops = (int)((distance / mpg) / gasTank);
    double finalTime = (distance / speed) + (gasStops * .5);

    Console.WriteLine("You'll have to stop for gas " + gasStops.ToString() + " time(s).");
    Console.WriteLine("With that in mind, it will take you " + finalTime.ToString("#.##") + " hours to get back to Manhattan!");
    Console.ReadKey();
}
```

2 - Beginning - Camping Trip

Problem Statement

You have been planning a three day camping trip with your family this Fall, but Kansas weather can be tricky! Looking at the weather forecast for the next week, find the warmest three consecutive days (*i.e. highest sum of three consecutive temperatures*) for your trip.

Input

Your program should take the following input:

- Seven temperatures (as whole numbers).

Output

Your program should output which days you decide to go camping on. Note that if there is a tie, you should always go camping on the later days.

Examples

Input	Output
Temperatures: **60 77 42 50 75 60 56**	Camping on days: **5, 6, 7**
Temperatures: **60 54 42 50 61 60 42**	Camping on days: **4, 5, 6**

```
days = 3
temps = [int(temp) for temp in input('Please enter the all 7 temperatures, separated by a space:').split(' ')]
max_temp = temps[0]
max_i = 0
i = 0

while i <= 7 - days:
    total = sum(temps[i:i+days])
    if total >= max_temp:
        max_temp = total
        max_i = i
    i += 1

print('Camping on days: {}'.format(', '.join([str(day) for day in range(max_i + 1, max_i + 1 + days)])))

```

3 Beginning — Martian Calendar
Problem Statement

This problem involves a Martian calendar, defined as follows. Each year is divided into 24 months. Months 6, 12, and 18 have 27 days, and all other months have 28 days.

Write a program that takes as input two integers giving a month and day, and produces as output the number of that date within the year, where 1/1 is numbered as day 1, and each following date is numbered successively. You may assume that the date is valid according to the above definition.

Example 1:

```
Enter month: 1
Enter day: 1

Day of year: 1
```

Example 2:

```
Enter month: 2
Enter day: 10

Day of year: 38
```

Example 3:

```
Enter month: 24
Enter day: 28

Day of year: 669
```

```
1  // 3 Beginning - Martian Calendar
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 MartianCalendarBeginning
10 {
11     class Program
12     {
13         static void Main(string[] args)
14         {
15             Console.Write("Enter month: ");
16             int month = Convert.ToInt32(Console.ReadLine());
17             Console.Write("Enter day: ");
18             int day = Convert.ToInt32(Console.ReadLine());
19             Console.WriteLine();
20             Console.Write("Day of year: ");
21             Console.WriteLine((month - 1) * 28 - (month - 1) / 6 + day);
22             Console.ReadLine();
23         }
24     }
25 }
26
```

4 Beginning - Tiling

Problem Statement

A room which is shaped like a rectangle needs to be tiled with equal-sized, square tiles. The pieces that are cut off to make the tiles fit cannot be used anywhere else. The tiling must start in a corner. There are two sizes of tiles available at different prices. Given the length and width of the room (in whole inches) and the sizes (in whole inches) and prices (dollars) of the two different tiles, determine which tile gives the lowest total cost. Display which tiles to use and the total cost for the tiles.

Example 1

Inputs:
Length of room: 600
Width of room: 400
Tile 1 length: 10
Tile 1 price: 0.40
Tile 2 length: 20
Tile 2 price: 0.60

Output:
The second tiles are cheapest at 360

Example 2

Inputs:
Length of room: 600
Width of room: 400
Tile 1 length: 18
Tile 1 price: 0.67
Tile 2 length: 12
Tile 2 price: 0.41

Output:
The first tiles are cheapest at 523.94

4 Beginning – Tiling
Solution

```
int main()
{
	int length, width, tile1size, tile2size;
	int Ntile1, Ntile2, x1, x2, y1, y2;
	float tile1price, tile2price, costtile1, costtile2;
	char q;
	q = 'Q';
	while (q == 'Q') {
		std::cout << "\nEnter length (in whole inches): ";
		std::cin >> length;
		std::cout << "\nEnter width (in whole inches): ";
		std::cin >> width;
		std::cout << "\nEnter tile 1 size (in whole inches): ";
		std::cin >> tile1size;
		std::cout << "\nEnter tile 1 price (dollars): ";
		std::cin >> tile1price;
		std::cout << "\nEnter tile 2 size (in whole inches): ";
		std::cin >> tile2size;
		std::cout << "\nEnter tile 2 price (dollars): ";
		std::cin >> tile2price;
		//tile1
		x1 = 1;
		y1 = length / tile1size;
		if (length == tile1size*y1) {
			x1 = 0;//if length is multiple of tilesize then no extra tile needed
		}
		x2 = 1;
		y2 = width / tile1size;
		if (length == tile1size*y1) {
			x2 = 0;//if width is multiple of tilesize then no extra tile needed
		}
		Ntile1 = (y1 + x1)*(y2 + x2);
		std::cout << "\nNum tile 1: " << Ntile1;
		costtile1 = Ntile1 * tile1price;
		//tile2
		x1 = 1;
		y1 = length / tile2size;
		if (length == tile2size*y1) {
			x1 = 0;//if length is multiple of tilesize then no extra tile needed
		}
		x2 = 1;
		y2 = width / tile2size;
		if (width == tile2size*y2) {
			x2 = 0;//if length is multiple of tilesize then no extra tileneeded
		}
		Ntile2 = (y2 + x2)*(y1 + x1);
		std::cout << "\nNum tile 2: " << Ntile2;
		costtile2 = Ntile2 * tile2price;
		//compare
		if (costtile1 < costtile2) { std::cout << "\nTile 1 costs less: "
			<< costtile1; }
		else {
			std::cout << "\nTile 2 costs the same or less: " << costtile2;
		}
		std::cin >> q;}  return 0;}
```

5 - Beginning - Partner Up

Problem Statement

You are hosting a holiday party. When you invited people to your party, you requested each guest to bring one partner so all of the party games would work out. However, you noticed that there were an odd number of people at the party so someone didn't bring a partner. Each of your invitations had a unique identifier (an integer) that your guests (and their partner) had to present to join the party. Find out which guest did not bring a partner.

Input

Your program should take the following two inputs in order:

- The number of people (guests and partners) at the party.
- The identifiers of each guest (and partner if applicable) from their invitation.

You may assume one guest did not bring a partner. You may also assume that each guest will be listed together with their partner if they have one.

Output

Your program should output the identifier of the guest that didn't bring a partner.

Examples

Input	Output
Attendance: **3** Guest ID(s): **2 5 5**	Guest **#2** did not bring a partner.
Attendance: **1** Guest ID(s): **10**	Guest **#10** did not bring a partner.
Attendance: **5** Guest ID(s): **3 3 2 2 7**	Guest **#7** did not bring a partner.

```
guests = int(input("Enter number of guests: "))
guest_list = [int(guest) for guest in input("Enter guest list: ").split(' ')]
seen = []
for i in range(guests):
    # make guest numbers
    if guest_list[i] not in seen and guest_list[i] not in guest_list[i+1:]:
        print('Guest #{} did not bring a partner.'.format(guest_list[i]))
    else:
        seen.append(guest_list[i])

```

6 Beginning — World Series
Problem Statement

Your favorite baseball team is playing in the World Series, but they may be in trouble — they must win all of the remaining games to win the series. The probability that they will win the series is the product of the probabilities of winning each of the remaining games. Your task is to write a program to calculate this probability. It should take as input a positive integer giving the number of games remaining, and for each of these games, a floating-point number giving the probability that your team will win that game. It should produce as output the probability that they win the World Series. You may assume that the number of games remaining is at least 1 and at most 4, and that each probability is at least 0 and at most 1.

Example 1:

```
Enter the number of games remaining: 1
Enter the probability of winning game 1: 0.45

The probability of winning the series is: 0.45
```

Example 2:

```
Enter the number of games remaining: 4
Enter the probability of winning game 1: 0.4
Enter the probability of winning game 2: 0.3
Enter the probability of winning game 3: 0.2
Enter the probability of winning game 4: 0.1

The probability of winning the series is: 0.0024
```

```
1  // 3 Beginning - World Series
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 WorldSeriesBeginning
10 {
11     class Program
12     {
13         static void Main(string[] args)
14         {
15             Console.Write("Enter the number of games remaining: ");
16             int n = Convert.ToInt32(Console.ReadLine());
17             double result = 1;
18             for (int i = 1; i <= n; i++)
19             {
20                 Console.Write("Enter the probability of winning game " + i + ":
                     ");
21                 double p = Convert.ToDouble(Console.ReadLine());
22                 result *= p;
23             }
24             Console.WriteLine();
25             Console.Write("The probability of winning the series is: ");
26             Console.WriteLine(result);
27             Console.ReadLine();
28         }
29     }
30 }
31
```