A1 Problem Statement

Occupied versus Unoccupied

Most houses have an electricity meter that records the amount of electricity that has been used since the meter was installed. This is typically recorded in kilowatt-hours. The amount of KwH shown on the meter is usually read at least monthly. The increase in the reading from one month to the next is what is used for billing the homeowner.

One homeowner is interested in knowing how much energy (average KwH per hour) is used when the house is un-occupied compared to when the house is occupied. So the owner records the reading each time the family leaves the house and then each time the family arrives back.

Assume that each record starts with an 'L' for leaving or an 'A' for arriving. The A and L records will alternate. Besides the meter reading, the record includes two more numbers that represents the day of the year (Jan 1 is day 1 and Dec 31 is day 365) and the hour of the day (24 hour clock). Ignore leap years. If the second date is numerically less than the first, it is assumed to be in the next year. An 'E' will end the list of records. The 'D' and 'H' representing day and hour are shown in the first example but do not appear in the record. Your program can read the whole line or prompt for each entry.

Given a set of 10 or less records, calculate the average kilowatt-hours used when the house is empty and the average when it is occupied.

Example 1

A 995 D 12 H 12
L 1200 D 14 H 14
A 1400 D 18 H 18
E
Answer: Occupied average is 4.1 KwH per hour; Empty average is 2 KwH per hour

Example 2

L 1800 360 12
A 1900 362 8
L 2200 364 10
A 2500 4 15
L 2800 6 20
A 3001 8 8
E
Answer: Occupied average is 5.82 KwH per hour; Empty average is 2.93 KwH per hour

A1 solution

```
#include <iostream>

int main()
{
	float EmptyKWH, EmptyH, OccKWH, OccH;
	int currentmeter, currentday, currenthour;
	int nextmeter, nextday, nexthour;
	char AorL, C;
	C = 'c';
	while (C == 'c')
	{
		EmptyKWH = 0; EmptyH = 0; OccKWH = 0; OccH = 0;
		std::cout << "\nStart Data: ";
		std::cin >> AorL;
		if (AorL == 'A' || AorL == 'L')
		{
			std::cin >> currentmeter;
			std::cin >> currentday;
			std::cin >> currenthour;
		}
		std::cin >> AorL;
		while (AorL != 'E')
		{
			std::cin >> nextmeter;
			std::cin >> nextday;
			std::cin >> nexthour;

			if (nextday < currentday) { nextday = nextday + 365; }

			if (AorL == 'A')
			{
				EmptyKWH = EmptyKWH + (nextmeter - currentmeter);
				EmptyH = EmptyH + (24 * (nextday - currentday) + (nexthour -
currenthour));
			}
			if (AorL == 'L')
			{
				OccKWH = OccKWH + (nextmeter - currentmeter);
				OccH = OccH + (24 * (nextday - currentday) + (nexthour -
currenthour));
			}
			currentmeter = nextmeter;
			currentday = nextday;
			if (currentday > 365) { currentday = currentday - 365; }
			currenthour = nexthour;
			std::cin >> AorL;
		}
		std::cout << "\nOccupied average is " << OccKWH / OccH;
		std::cout << "\nEmpty average is " << EmptyKWH / EmptyH;
		std::cout << "\n enter c to continue ";
		std::cin >> C;
	}
    return 0;
}
```

A1 Test Cases – Electricity Usage

The output should be real numbers with decimal values. You can tell them the input values. Formatting of the inputs or outputs is not important.

First attempt – try the following 2 test cases. You can tell the team which were incorrect

Test 1:

Inputs	A	995	12	12
	L	1200	14	14
	A	1400	18	18
	E			

Required output: OCC 4.1 EMPTY 2

Test 2:

Inputs	L	3000	60	12
	A	4000	70	12
	L	6000	100	14
	E			

Required output OCC 2.77 EMPTY 4.167

Second attempt – repeat the above 3 tests and the following two tests

Test 4:

Inputs	A	2000	12	8
	L	4000	16	20
	A	6000	26	10
	E			

Required output OCC 18.51 EMP 8.69

Test 5:

Inputs	L	1800	360	12
	A	1900	362	8
	L	2200	364	10
	A	2500	4	15
	L	2800	6	20
	A	3001	8	8
	E			

Required output OCC 5.82 EMPTY 2.93

2 Advanced — Logarithms

The *natural logarithm* of x, or $\ln x$, is an important function in calculus. For $0 < x < 2$, $\ln x$ can be approximated using the following formula:

$$\ln x \approx (x-1)\left(\frac{1}{1} - (x-1)\left(\frac{1}{2} - (x-1)\left(\frac{1}{3} - \cdots - (x-1)\left(\frac{1}{n}\right)\cdots\right)\right)\right)$$

A larger value of n will produce a more accurate approximation. For $x \geq 2$, by repeatedly dividing x by any c such that $1 < c < 2$, we can obtain y such that $0 < y < 2$. Then $x = c^k y$, where k is the number of times x was divided by c. We then have

$$\ln x = k \ln c + \ln y,$$

and $\ln c$ and $\ln y$ can be approximated using the above formula.

Write a program that takes as input x, c, and n and produces as output the approximation of $\ln x$ given by the above formula with the given values of c and n. When computing y, use only as many divisions by c as are necessary to get $y < 2$. You may assume that $x > 0$ ($\ln x$ is undefined otherwise), that $1 < c < 2$, and that n is a positive integer no more than 1000. Note that your goal is *not* to calculate $\ln x$, but to see what approximation of $\ln x$ is produced by the given c and n. Your results should agree with the examples below on at least the first 3 digits.

Hint: When doing the divisions, be sure you are doing floating-point division, not integer division.

Example 1:

```
Enter x: 1000
Enter c: 1.8
Enter n: 5

ln x = 7.19727277221476
```

Example 2:

```
Enter x: 0.5
Enter c: 1.1
Enter n: 1000

ln x = -0.693147180559945
```

Example 3:

```
Enter x: 10
Enter c: 1.2
Enter n: 1

ln x = 2.73806699467815
```

2 Advanced — Logarithms
Test Cases

For each test case, check only the first 3 significant digits of the output.

Test Case 1:

```
Enter x: 10000
Enter c: 1.5
Enter n: 5

ln x = 9.25071289887356
```

Test Case 2:

```
Enter x: 0.5
Enter c: 1.8
Enter n: 1000

ln x = -0.693147180559945
```

Test Case 3:

```
Enter x: 50
Enter c: 1.25
Enter n: 1

ln x = 4.5092186044416
```

Second Submission: Do the above tests, plus the following:

Test Case 4:

```
Enter x: 15
Enter c: 1.6
Enter n: 20

ln x = 2.70804687843129
```

```
// 2 Advanced - Logarithms

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ksu.Hspc2015.Logarithms
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter x: ");
            double x = Convert.ToDouble(Console.ReadLine());
            Console.Write("Enter c: ");
            double c = Convert.ToDouble(Console.ReadLine());
            Console.Write("Enter n: ");
            int n = Convert.ToInt32(Console.ReadLine());
            int k = 0;
            while (x >= 2)
            {
                x = x / c;
                k++;
            }
            Console.WriteLine();
            Console.WriteLine("ln x = " + (Log(x, n) + k * Log(c, n)));
            Console.ReadLine();
        }

        private static double Log(double x, int n)
        {
            x -= 1;
            double result = 0;
            for (int i = n; i > 0; i--)
            {
                result = x * (1.0 / i - result);
            }
            return result;
        }
    }
}
```

A3 Problem Statement

Making change with quarters, nickels, and pennies

Given a number, n, of coins and an amount, X, determine what amounts between 0 and X can be made with exactly n coins.

Note: we won't use dimes because that will make the problem much more difficult. We will also limit the number of coins to less than 5 for similar reasons.

Example 1:

n = 1 and X = 100

Answer: 1; 5; 25

Example 2:

n = 3 and X =50

Answer: 3; 7; 11; 15; 27; 31; 35

A3 solution

```
#include <iostream>

int main()
{
	int n, X;
	int Q, N, P, T, j, I;
	char C;
	C = 'c';
	while (C == 'c')
	{

		std::cout << "\nEnter number of coins: ";
		std::cin >> n;
		std::cout << "\nEnter amount: ";
		std::cin >> X;

		for (j = 1; j < X + 1; j++) {
			T = j;
			Q = 0; N = 0; P = 0; I = 0;
			while (I < n && T > 0)
			{
				while (T >= 25) { T = T - 25; I++; Q++; }
				while (T >= 5) { T = T - 5; I++; N++; }
				while (T >= 1) { T = T - 1; I++; P++; }
			}
			if ((j == 25 * Q + 5 * N + P) && I == n)
			{
				std::cout << "\n " << j << " using " << Q << " quarters "
					<< N << " nickles " << P << " pennies";
			}

		}

		std::cin >> C;
	}
	return 0;
}
```

A3 Test Cases – Change

The output should be integers. You can tell them the input values. Formatting of the inputs or outputs is not important.

First attempt – try the following test cases. You can tell the team which were incorrect

Test 1:

Inputs	n = 3	X = 50							
Required output		3	7	11	15	27	31	35	

Test 2:

Inputs	n = 4	X = 40							
Required output		4	8	12	16	20	28	32	36	40

Second attempt – repeat the above 2 tests and the following two tests

Test 3:

Inputs	n=3	X = 100								
Required output 75		3	7	11	15	27	31	35	51	55

Test 4:

Inputs	n=1	X = 25		
Required output		1	5	25

4 Advanced — The Josephus Problem

The *Josephus problem* can be stated as follows. n people are seated in a circle and numbered clockwise from 0 to $n-1$. Starting with the person numbered 0, they count off, starting at 1 and going clockwise. Each time the count reaches k, the person counting k is removed from the circle, and the counting restarts at 1 with the next person in the clockwise direction. The question is to determine which person will be the last remaining. For example, if $n = 7$ and $k = 3$, persons 2 and 5 are the first to be removed. Then the count goes to 6, 0, and 1, so 1 is removed. Because 2 and 5 have already been removed, 6 is the next to be removed. Then because 1 and 2 have been removed, 4 is removed next. At this point, only 0 and 3 remain, so 0 is removed next, leaving 3.

Write a program to input n and k and output the last person remaining. You may assume that n and k are both positive integers.

Example 1:

```
Enter n: 7
Enter k: 3

Person remaining: 3
```

Example 2:

```
Enter n: 13
Enter k: 21

Person remaining: 0
```

Example 3:

```
Enter n: 20
Enter k: 5

Person remaining: 6
```

4 Advanced — The Josephus Problem
Test Cases

Test Case 1:

```
Enter n: 74
Enter k: 15

Person remaining: 4
```

Test Case 2:

```
Enter n: 23
Enter k: 37

Person remaining: 7
```

Test Case 3:

```
Enter n: 15
Enter k: 1

Person remaining: 14
```

Second Submission: Do the above tests, plus the following:

Test Case 4:

```
Enter n: 123
Enter k: 32

Person remaining: 104
```

```
// 4 Advanced - The Josephus Problem

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ksu.Hspc2015.Josephus
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter n: ");
            int n = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter k: ");
            int k = Convert.ToInt32(Console.ReadLine());
            int remaining = n;
            bool[] removed = new bool[n];
            int i = 0;
            int step = 0;
            while (true)
            {
                if (!removed[i])
                {
                    if (remaining == 1)
                    {
                        break;
                    }
                    step++;
                    if (step == k)
                    {
                        removed[i] = true;
                        remaining--;
                        step = 0;
                    }
                }
                i++;
                if (i == n)
                {
                    i = 0;
                }
            }
            Console.WriteLine();
            Console.WriteLine("Person remaining: " + i);
            Console.ReadLine();
        }
    }
}
```

A5 Problem Statement

Probability of a value occurring on a toss of 4 dice

Given 4 normal dice and an integer X, calculate the probability of X being the sum of a roll of the 4 dice.

For example, with 2 dice, if X = 7, there are six combinations of the faces of the two dice (1,6; 2,5; 3,4; 4,3; 5,2; and 6,1). The probability of the sum being 7 is 6/(6*6).

Example 1:

X = 5

answer is 0.003086

A5 – Solution

```
#include "stdafx.h"
#include <iostream>

int main()
{
	int X, I, J, K, L, N;
	std::cout << "\nEnter X:";
	std::cin >> X;
	N = 0;
	if (X < 4 || X > 24) { std::cout << "\n Not possible"; return 0; }
	for (I = 6; I > 0; I--) {
		for (J = 6; J > 0; J--) {
			for (K = 6; K > 0; K--) {
				for (L = 6; L > 0; L--) {
					if (I + J + K + L == X) {
						std::cout << "\nAnswer: " << I << " " << J
							<< " " << K << " " << L;
						N++;
					}
				}
			}
		}
	}
	std::cout << "\n "<< N << "  solutions";
	std::cout << " probabilty is " << N / (6.0 * 6.0 * 6.0 * 6.0);
    return 0;
```

A5 Test Cases – Probabilities.

First attempt – try the following test cases. You can tell the team which were incorrect

Test 1:

Inputs x = 5

Required output 0.003086

Test 2:

Inputs x = 23

Required output 0.003086

Test 3:

Inputs X = 12

Required output 0.09645

Second attempt – repeat the above 3 tests and the following two tests

Test 1:

Inputs X = 9

Required output 0.0432

Test 4:

Inputs X = 2

Required output not possible

6 Advanced — Multiplier

A certain mathematical system contains three elements, a, b, and c, which can be multiplied using the following multiplication table:

	a	b	c
a	a	c	b
b	c	a	a
c	b	c	b

Thus, to determine the result of the multiplication ab, we look in the row for a and the column for b, and see that the result is c. A more complicated multiplication such as $ab(ca)$ can be done a single multiplication at a time:

1. $ab = c$
2. $ca = b$
3. $cb = c$

Write a program that reads in a string containing an expression to be multiplied and produces the result of the multiplication. You may assume that the string is nonempty, contains only the characters, a, b, c, (, and), and has length at most 30. Furthermore, you may assume that the parentheses are balanced, and that each pair encloses a nonempty expression.

Example 1:

```
Enter expression: ab(ca)

Result = c
```

Example 2:

```
Enter expression: b

Result = b
```

Example 3:

```
Enter expression: ((ab)c(ca)((b)))c

Result = b
```

6 Advanced — Multiplier
Test Cases

Test Case 1:

```
Enter expression: a(b(cb(a)b(cb)))

Result = a
```

Test Case 2:

```
Enter expression: c

Result = c
```

Test Case 3:

```
Enter expression: abab(cba(bc)(ac)c)

Result = c
```

Second Submission: Do the above tests, plus the following:

Test Case 4:

```
Enter expression: b(c(a(b)))(abc)(cba)

Result = c
```

```
// 6 Advanced - Multiplier

using System;

namespace Ksu.Hspc2015.AdvancedMultiplier
{
    class Program
    {
        private static char[,] _results =
        {
            {'a', 'c', 'b' },
            {'c', 'a', 'a' },
            {'b', 'c', 'b' }
        };

        private static string _expression;

        private static int _currentPosition = 0;

        static void Main(string[] args)
        {
            Console.Write("Enter expression: ");
            _expression = Console.ReadLine();
            Console.WriteLine();
            Console.WriteLine("Result = " + Evaluate());
            Console.ReadLine();
        }

        private static char Evaluate()
        {
            char result;
            if (_expression[_currentPosition] == '(')
            {
                _currentPosition++;
                result = Evaluate();
            }
            else
            {
                result = _expression[_currentPosition];
            }
            _currentPosition++;
            while (_currentPosition < _expression.Length && _expression[_currentPosition] != ')')
            {
                if (_expression[_currentPosition] == '(')
                {
                    _currentPosition++;
                    result = _results[result - 'a', Evaluate() - 'a'];
                }
                else
                {
                    result = _results[result - 'a', _expression[_currentPosition] - 'a'];
                }
                _currentPosition++;
            }
            return result;
        }
    }
}
```