B1 Problem Statement

Average electricity usage

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 (Kwh). 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.

Assume that each reading is associated with a number that represents the day of the year (Jan 1 is day 1 and Dec 31 is day 365). Ignore leap years. If the second date is numerically less than the first, it is assumed to be in the next year. Assume that the readings are taken about the same time each day.

Given two readings, calculate the average kilowatt-hours used per day between those readings.

Example 1

Day 10 Reading 1000
Day 12 Reading 1043

Average usage is 21.5 kilowatt-hours

Example 2

Day 360 Reading 1099
Day 10 Reading 1200

Average usage is 6.73
B1 Solution

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

int main()
{
    float R1, R2;
    int D1, D2;
    char C;
    C = 'c';
    while (C == 'c')
    {
        std::cout << "Enter First Day:";
        std::cin >> D1;
        std::cout << "Enter First Reading:";
        std::cin >> R1;
        std::cout << "Enter Second Day:";
        std::cin >> D2;
        std::cout << "Enter Second Day:";
        std::cin >> R2;
        if (D2 < D1) { D2 = D2 + 365; }
        std::cout << "Average usage is " << (R2 - R1) / (D2 - D1) << " Kilowatt-Hr/Day";
        std::cin >> C;
    }

    return 0;
}
B1 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 value is day. 2\textsuperscript{nd} is meter reading.

\textbf{First attempt – try the following test cases.}

You can tell the team which were incorrect

Test 1:

\begin{verbatim}
Inputs  360  1099
      10  1200
Required output  6.73
\end{verbatim}

Test 2:

\begin{verbatim}
Inputs  10  1200
      15  1700
Required output  100
\end{verbatim}

Test 3:

\begin{verbatim}
Inputs  25  2000
      45  4444
Required output  122.2
\end{verbatim}

\textbf{Second attempt – repeat the above 3 tests and the following two tests}

Test 1:

\begin{verbatim}
Inputs  300  3333
      35  6666
Required output  33.33
\end{verbatim}

Test 1:

\begin{verbatim}
Inputs  10  1000
      12  1043
Required output  21.5
\end{verbatim}
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. Write a program that takes as input $x$ and $n$ and produces as output the approximation of $\ln x$ given by the above formula with the given value of $n$. You may assume that $0 < x < 2$ 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 $n$. 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. Also, note that the innermost parentheses are around $1/n$. Your calculation will need to start there and work its way to the left.

**Example 1:**

Enter $x$: 1.8  
Enter $n$: 10  
$\ln x = 0.583275456934603$

**Example 2:**

Enter $x$: 0.001  
Enter $n$: 1  
$\ln x = -0.999$

**Example 3:**

Enter $x$: 1.1  
Enter $n$: 1000  
$\ln x = 0.0953101798043249$
For each test case, check only the first 3 significant digits of the output.

Test Case 1:

Enter x: 1.9
Enter n: 10

ln x = 0.626

Test Case 2:

Enter x: 1.1
Enter n: 1

ln x = 0.1

Test Case 3:

Enter x: 0.5
Enter n: 1000

ln x = -0.693

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

Test Case 4:

Enter x: 0.9
Enter n: 20

ln x = -0.105
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ksu.Hspc2015.BeginningLogarithms
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter x: ");
            double x = Convert.ToDouble(Console.ReadLine()) - 1;
            Console.Write("Enter n: ");
            int n = Convert.ToInt32(Console.ReadLine());
            double result = 0;
            for (int i = n; i > 0; i--)
            {
                result = x * (1.0 / i - result);
            }
            Console.WriteLine();
            Console.WriteLine("ln x = "+ result);
            Console.ReadLine();
        }
    }
}
B3 Problem Statement

Making change with quarters, nickels, and pennies

Given a number, n, of coins and an amount, X, decide whether or not you can make that amount of change with exactly n coins. Show what the coins will be.

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 = 10

Answer: yes  2 nickels

Example 2:

n = 3 and X =10

Answer: not possible
```cpp
#include "stdafx.h"
#include <iostream>

int main()
{
    int n, X;
    int Q, N, P, T, I;
    char C;
    C = 'c';
    while (C == 'c')
    {
        Q = 0; N = 0; P = 0; I = 0;
        std::cout << "Enter number of coins: ";
        std::cin >> n;
        std::cout << "Enter amount: ";
        std::cin >> X;
        T = X;
        while (I < n && T > 0)
        {
            while (T >= 25) { T = T - 25; I++; Q++; std::cout << "Q" << Q; }
            while (T >= 5) { T = T - 5; I++; N++; std::cout << "N" << N; }
            while (T >= 1) { T = T - 1; I++; P++; std::cout << "P" << P; }
        }
        if ((X == 25 * Q + 5 * N + P) && I == n)
        {
            std::cout << "
" << Q << " quarters " << N << " nickles " << P << " pennies";
        }
        else
        {
            std::cout << "\n not possible";
        }
        std::cin >> C;
    }
    return 0;
}
```
B3 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 = 4 X = 100
Required output yes 4 quarters

Test 2:
Inputs n = 4 X = 99
Required output not possible

Test 3:
Inputs n = 4 X = 12
Required output yes 2 nickels 2 pennies

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

Test 4:
Inputs n = 3 X = 31
Required output yes, 1 quarter, 1 nickel, 1 penny

Test 1:
Inputs n = 3 X = 2
Required output not possible
Write a program that reads positive integer values \( k \) and \( n \) and produces as output all nonnegative integer multiples of \( k \) that are less than \( n \). Note that 0 is a nonnegative integer multiple of any integer. Your multiples must be displayed in increasing order. You may assume that both \( k \) and \( n \) are positive integers.

**Example 1:**

Enter \( k \): 2  
Enter \( n \): 10  

0  
2  
4  
6  
8  

**Example 2:**

Enter \( k \): 100  
Enter \( n \): 10  

0  

**Example 3:**

Enter \( k \): 1000  
Enter \( n \): 2500  

0  

1000  

2000
Test Case 1:

Enter k: 1
Enter n: 5

0
1
2
3
4

Test Case 2:

Enter k: 10
Enter n: 5

0

Test Case 3:

Enter k: 7
Enter n: 30

0
7
14
21
28

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

Test Case 4:

Enter k: 4
Enter n: 35

0
4
8
12
16
20
24
28
32
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Ksu.Hspc2015.Multiples{
    class Program{
        static void Main(string[] args){
            Console.Write("Enter k: ");
            int k = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter n: ");
            int n = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine();
            for (int i = 0; i < n; i += k){
                Console.WriteLine(i);
            }
        }
    }
}
B5 Problem Statement

Dice

The usual die (one “dice”) has 6 faces and the numbers 1 through 6 appearing on those faces. If you roll a normal die multiple times and record the values on the faces, the average of all the rolls should be 3.5. This is called the expected value. It is the sum of all the faces divided by the number of faces, that is $1 + 2 + 3 + 4 + 5 + 6$ divided by 6.

Given a die with $n$ sides with a lowest value of $x$, calculate the expected value. Assume that each face is equally likely and that the values increase by 1. For example, if $n$ is 4 (a pyramid) and $x$ is 2, the sides are 2,3,4,5 and the expected value is $14/4 = 3.5$.

Example 1:

$n = 7$ and $x$ is 10

answer is 13
B5 solution

#include <iostream>

int main()
{
    int n, X;
    int j;
    float sum;
    char C;
    C = 'c';
    while (C == 'c')
    {
        std::cout << "Enter number of sides: ";
        std::cin >> n;
        std::cout << "Enter start value: ";
        std::cin >> X;
        sum = 0;
        for (j = X; j < n+X; j++) {
            sum = sum + j;
            std::cout << " \n " << j << " " << sum;
        }
        std::cout << "Expected value is " << sum / n;
        std::cin >> C;
    }
    return 0;
}
B5 Test Cases – Expected value of dice

The output should be reals. 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 = 7; X = 10
Required output: 13

Test 2:
Inputs: n = 6; X = 1
Required output: 3.5

Test 3:
Inputs: n = 4; X = 4
Required output: 5.5

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

Test 1:
Inputs: n = 11; X = 5
Required output: 10

Test 1:
Inputs: n = 20; X = 40
Required output: 49.5
6 Beginning — Multiplier

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

<table>
<thead>
<tr>
<th></th>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>$a$</td>
<td>$c$</td>
<td>$b$</td>
</tr>
<tr>
<td>$b$</td>
<td>$c$</td>
<td>$a$</td>
<td>$a$</td>
</tr>
<tr>
<td>$c$</td>
<td>$b$</td>
<td>$c$</td>
<td>$b$</td>
</tr>
</tbody>
</table>

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 $abca$ can be done a single multiplication at a time:

1. $ab = c$
2. $cc = b$
3. $ba = c$

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, $a$, $b$, and $c$, and has length at most 20.

**Example 1:**

Enter expression: abca
Result = c

**Example 2:**

Enter expression: b
Result = b

**Example 3:**

Enter expression: aabbccbbaa
Result = c
Test Case 1:

Enter expression: abcabc
Result = b

Test Case 2:

Enter expression: c
Result = c

Test Case 3:

Enter expression: abcbaabcbaabcbaabcba
Result = a

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

Test Case 4:

Enter expression: aaacccb
Result = a
using System;

namespace Ksu.Hspc2015.BeginningMultiplier
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter expression: ");
            string expr = Console.ReadLine();
            char result = expr[0];
            for (int i = 1; i < expr.Length; i++)
            {
                if (result == 'a')
                {
                    if (expr[i] == 'a')
                    {
                        result = 'a';
                    }
                    else if (expr[i] == 'b')
                    {
                        result = 'c';
                    }
                    else
                    {
                        result = 'b';
                    }
                }
                else if (result == 'b')
                {
                    if (expr[i] == 'a')
                    {
                        result = 'c';
                    }
                    else
                    {
                        result = 'a';
                    }
                }
                else
                {
                    if (expr[i] == 'b')
                    {
                        result = 'c';
                    }
                    else
                    {
                        result = 'b';
                    }
                }
            }
            Console.WriteLine();
            Console.WriteLine("Result = " + result);
            Console.ReadLine();
        }
    }
}