1 Beginning — Area Between Lines

A line in the $x/y$ plane can be described by an equation of the form

$$y = mx + b$$

(1)

where $m$ denotes the slope, and $b$ denotes the $y$-intercept. Write a program that reads the slope and $y$-intercept of a line and outputs the area of the triangle bounded by this line and the $x$- and $y$-axes, as shown in the figure below. The area of this triangle is given by $bh/2$, where $b$ (the base) is the horizontal distance from the intersection of the given line with $x$-axis to the intersection of the two axes, and $h$ (the height) is the vertical distance from the intersection of the two axes to the intersection of the given line with the $y$-axis. You may assume that both the slope and the $y$-intercept are strictly positive. As a result, the height will simply be the $y$-intercept, and the base will be the negative of the $x$-value that makes $y = 0$.

![Diagram of a triangle formed by a line and the x and y axes](image)

Example 1:

Enter slope: 0.75
Enter $y$-intercept: 3
Area = 6

Example 2:

Enter slope: 1
Enter $y$-intercept: 2.4
Area = 2.88
Note: Results should be accurate to 3 significant digits.

Test Case 1:

Enter slope: 0.3
Enter y-intercept: 4.6
Area = 35.26666

Test Case 2:

Enter slope: 2.7
Enter y-intercept: 1.8
Area = 0.6

Retest: Do the above tests, plus the following:

Test Case 3:

Enter slope: 3.3
Enter y-intercept: 10.7
Area = 17.34697
using System;

namespace Contest2008.AreaBeginning{

class Area{
    
    /// Reads a slope and y-intercept, both of which must
    /// be positive, and prints the area bounded by the
    /// line so described and the x- and y-axes.
    ///
    static void Main(string[] args)
    {
        Console.Write("Enter slope: ");
        float slope = Single.Parse(Console.ReadLine());
        Console.Write("Enter y-intercept: ");
        float yIntercept = Single.Parse(Console.ReadLine());
        float baseLen = yIntercept / slope;
        float area = baseLen * yIntercept / 2;
        Console.WriteLine("Area = " + area);
        Console.ReadLine();
    }
}
}
Problem 2 Beginning

Simulation of water flowing into tank

In this task, you will write a discrete simulation of the flow of water into a tank that has holes in the side of the tank. You will prompt for the height and diameter of the tank, for the rate of inflow into the tank and for the length of time of the simulation. All measurements will be metric. The height and diameter are in centimeters. The flows are in cubic centimeters per second. Each step of the simulation will be a second. The answer will be the height of the water in the tank at the end of the simulation. If the water overflows the top of the tank, a message will be displayed and the simulation will stop.

The holes in the side of the tank will be at each multiple of 5cm in height. The water will be assumed to start flowing through the hole in the simulation step after the water height reaches or exceeds the height of the hole. The flow through each hole will be 30 cc/sec.

The formula for the area of the bottom of a circular tank is \( \pi \times (\text{diameter}/2) \times (\text{diameter}/2) \times \text{height} \). \( \pi \) will be estimated as 3.14159

<table>
<thead>
<tr>
<th>Example 1:</th>
<th>Example 2:</th>
<th>Example 3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Height: 100.0</td>
<td>Height: 100.0</td>
<td>Height: 10.0</td>
</tr>
<tr>
<td>Diameter: 10.0</td>
<td>Diameter: 10.5</td>
<td>Diameter: 7.5</td>
</tr>
<tr>
<td>Inflow rate: 50.0</td>
<td>Inflow rate: 150.0</td>
<td>Inflow rate: 100.0</td>
</tr>
<tr>
<td>Time: 20</td>
<td>Time: 20</td>
<td>Time: 20</td>
</tr>
<tr>
<td>Height is 8.4</td>
<td>Height is 21.13</td>
<td>Water is over the top in 5 seconds</td>
</tr>
</tbody>
</table>
Tests 2 Beginning  
Simulation of water flowing into tank

Don’t grade on formatting. Also, on this problem, don’t worry about the values to the right of the decimal point.

**First round testing**

Example 1:  
Height: 100.0  
Diameter: 10.0  
Inflow rate: 50.0  
Time: 20  
Height is 8.4

Example 2:  
Height: 30.0  
Diameter: 8.5  
Inflow rate: 45.0  
Time: 20  
Height is 9.25

Example 3:  
Height: 10.0  
Diameter: 7.5  
Inflow rate: 100.0  
Time: 20  
Water is over the top in 5 seconds

**Second round testing – do all the above plus the following tests**

Example 4:  
Height: 20.0  
Diameter: 8.5  
Inflow rate: 150.0  
Time: 20  
Over the top in 11 seconds

Example 5:  
Height: 10.0  
Diameter: 9.5  
Inflow rate: 90.0  
Time: 20  
Water is over the top in 9 seconds

Example 6:
Solution 2 Beginning – Simulation of water tank

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

using namespace std;
using namespace System;

int main(array<System::String ^> ^args)
{
    double inflow, height, dia;
    int time, i, numholes;
    double waterlevel, deltalevel;
    int holeflow = 30;

    cout<<"enter height of container in centimeters:"; cin >> height;
    cout<<"enter diameter of container in centimeters:"; cin >> dia;
    cout<<"enter inflow of water in cubic centimeters / sec:";
    cin >> inflow;
    cout<<"enter time to run simulation in seconds:"; cin >> time;

    waterlevel = 0.0;
    for(i=0;i<time+1;i++)
    {
        numholes = (int) waterlevel/5; cout<<"\nholes "<<numholes;
        deltalevel = (inflow - holeflow*numholes)/(3.14159*dia*dia/4);
        if (deltalevel < 0) deltalevel = 0;
        waterlevel = waterlevel + deltalevel;
        cout<<"\n waterlevel "<<waterlevel
        if (waterlevel > height) {
            cout<< "\n water is over the top.
 stop simulation at time: "<i;
            return 0;
        }
    }
    cout<< "\n water height is "< waterlevel;
    return 0;
}
Write a program that reads an integer \( n \), \( 1 \leq n \leq 999,999,999 \), and prints each of its digits, from left to right, as an English word. Your input must be a single number.

**Example 1:**

Enter \( n \): 123456789
one
two
three
four
five
six
seven
eight
nine

**Example 2:**

Enter \( n \): 10
one
zero
Note: Input must be entered as a single number.

Test Case 1:

Enter n: 876543210
eight
seven
six
five
four
three	
two
one
zero

Test Case 2:

Enter n: 1
one

Test Case 3:

Enter n: 999
nine
nine
nine

Restest: Use the above tests, plus the following.

Test Case 4:

Enter n: 66506
six
six
five
zero
six
namespace Contest2008.NumberSpellingBeginning
{

class Speller
{
    /// Reads in a positive integer containing at most 9 digits, and prints each digit as an English word.
    ///
    /// static void Main(string[] args)
    ///
    /// A string is easier to work with, so no sense converting it to an int.
    string number = Console.ReadLine();

    /// A foreach could be used, but this is easier to translate to other languages.
    for (int i = 0; i < number.Length; i++)
    {
        if (number[i] == '0')
            Console.WriteLine("zero");
        if (number[i] == '1')
            Console.WriteLine("one");
        if (number[i] == '2')
            Console.WriteLine("two");
        if (number[i] == '3')
            Console.WriteLine("three");
        if (number[i] == '4')
            Console.WriteLine("four");
        if (number[i] == '5')
            Console.WriteLine("five");
        if (number[i] == '6')
            Console.WriteLine("six");
        if (number[i] == '7')
            Console.WriteLine("seven");
        if (number[i] == '8')
            Console.WriteLine("eight");
        if (number[i] == '9')
            Console.WriteLine("nine");
    }
}

Console.ReadLine();
}
Problem 4 Beginning

Numbers of Fridays in a month

This program will assume that the month has 30 days. The program will accept a character that specifies the day of the week of the first day of the month and a second char that is the day of interest. The program will respond with how many times that day of interest occurs in the month. The days will be represented by S (Sunday), M (Monday), T (Tuesday), W (Wednesday), U (Thursday), F (Friday) and A (Saturday).

Example 1:
Enter day of the week of the first day of the month: W
Enter day of interest: U
Such days in the month: 5

Example 2:
Enter day of the week of the first day of the month: M
Enter day of interest: F
Such days in the month: 4
Tests 4 Beginning Numbers of Fridays in a month

**First round testing**

Test 1:
- Enter day of the first day: W
- Enter day of interest: U
- Such days in the month: 5

Test 2:
- Enter day of the first day: M
- Enter day of interest: S
- Such days in the month: 4

**Second round of testing – do all the above plus the tests below**

Test 3:
- Enter day of the first day: U
- Enter day of interest: U
- Such days in the month: 5
Solution 4 Beginning - Calendar

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

using namespace std;
using namespace System;

int main(array<System::String ^> ^args)
{
    int d; char(firstday; int f; char day; int y;
    int n = 0; int i;
    cout << "\n enter number of days in the month: "); cin >> d;
    cout << "\n enter the day of the week of the first day: "
    cin >> firstday;
    cout << "\n enter the day of interest: "); cin >> day;

    if (day == 's' || day == 'S') { y = 0;)
    if (day == 'm' || day == 'M') { y = 1;)
    if (day == 't' || day == 'T') { y = 2;)
    if (day == 'w' || day == 'W') { y = 3;)
    if (day == 'u' || day == 'U') { y = 4;)
    if (day == 'f' || day == 'F') { y = 5;)
    if (day == 'a' || day == 'A') { y = 6;)

    if (firstday == 's' || firstday == 'S') { f = 0;)
    if (firstday == 'm' || firstday == 'M') { f = 1;)
    if (firstday == 't' || firstday == 'T') { f = 2;)
    if (firstday == 'w' || firstday == 'W') { f = 3;)
    if (firstday == 'u' || firstday == 'U') { f = 4;)
    if (firstday == 'f' || firstday == 'F') { f = 5;)
    if (firstday == 'a' || firstday == 'A') { f = 6;)

    if (f < y) {d = d - (y - f);}
else {d = d - (f - y);)
    for (i = 1; i*7 < d; i++) {
        n++;
    }
    cout << n + 1;
}
}
There is a rectangular box on the floor and a right circular cone on top of the box, as shown in the figure below. You are to write a program that takes as input the length, width, and height of the box, and the slant height and radius of the cone, and produces as output the surface area, excluding the face that is on the floor, the circular portion of the box covered by the cone, and the circular face of the cone. You may assume that all input values are strictly positive, and that the length and width of the box are both at least twice the radius of the cone. The following formulas may be useful:

- Area of a rectangle: length \times width
- Area of a circle: \pi \times radius^2
- Surface area of a cone, excluding its circular face: \pi \times radius \times slant height.

Use 3.1416 to approximate \pi.

Example 1:

Enter length of box: 4
Enter width of box: 5
Enter height of box: 2
Enter slant height of cone: 3
Enter radius of cone: 1
Area = 62.2832

Example 2:

Enter length of box: 6.7
Enter width of box: 5.2
Enter height of box: 3.3
Enter slant height of cone: 7.6
Enter radius of cone: 1.4
Area = 140.6491
Note: Results should be accurate to 3 significant digits.

Test Case 1:

Enter length of box: 1.2
Enter width of box: 3.4
Enter height of box: 5.6
Enter slant height of cone: 7.8
Enter radius of cone: 0.5
Area = 67.06684

Test Case 2:

Enter length of box: 9.8
Enter width of box: 7.6
Enter height of box: 5.4
Enter slant height of cone: 3.2
Enter radius of cone: 1
Area = 269.3115

Retest: Do the above tests, plus the following:

Test Case 3:

Enter length of box: 5
Enter width of box: 5
Enter height of box: 5
Enter slant height of cone: 5
Enter radius of cone: 2
Area = 143.8496
/ * Beginning 5 - Surface Area *
 */
using System;

namespace Contest2008.SurfaceAreaBeginning
{
    class SurfaceArea
    {
        const float Pi = 3.1416F;

        static void Main(string[] args)
        {
            Console.WriteLine("Enter length of box: ");
            float length = Single.Parse(Console.ReadLine());
            Console.WriteLine("Enter width of box: ");
            float width = Single.Parse(Console.ReadLine());
            Console.WriteLine("Enter height of box: ");
            float height = Single.Parse(Console.ReadLine());
            Console.WriteLine("Enter slant height of cone: ");
            float slantHeight = Single.Parse(Console.ReadLine());
            Console.WriteLine("Enter radius of cone: ");
            float radius = Single.Parse(Console.ReadLine());
            float boxArea = 2 * (length * height + width * height) + length * width;
            float circleArea = Pi * radius * radius;
            float coneArea = Pi * radius * slantHeight;
            float area = boxArea - circleArea + coneArea;
            Console.WriteLine("Area = "+area);
            Console.ReadLine();
        }
    }
}
Problem 6 Beginning    Sudoku Validation

Sudoku is a Japanese numerical problem. The game is played on a 9 by 9 array. The solution is an array such that every column has exactly one instance of the integers from 1 to 9. Every row has exactly one instance of the integers from 1 to 9 and each of the 3 by 3 sub-arrays has exactly one instance of the integers from 1 to 9. The program will accept a row of a Sudoku problem. It will validate that the row has exactly one instance of each of the integers from 1 to 9.

Example 1:
Enter: 1 2 3 4 5 6 7 8 9
Output: correct

Example 2:
Enter: 1 2 2 3 4 5 6 7 8
Output: incorrect
Test 6 Beginning                        Sudoku Validation

**Round 1 – format is not important**

Test 1:

Enter: 1 2 3 4 5 6 7 8 9  
Output: correct

Test 2:  

Enter: 1 2 2 3 4 5 6 7 8  
Output: incorrect

Test 3:  

Enter: 9 8 8 7 6 5 3 3 1  
Output is incorrect

**Round 2 – try all the above tests and the following tests**

Test 4:  

Enter: 0 1 2 3 4 5 6 7 9 9  
Output: incorrect

Test 5:  

Enter: 9 8 7 4 5 6 3 2 1  
Output: correct

Test 6:  

Enter: 7 8 9 1 2 3 6 5 4  
Output: correct
Solution 6 Beginning - Sudoku Validation

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

using namespace std;
using namespace System;

int main(array<System::String ^> ^args)
{
    int check[9], i, val;
    for(i=0; i<9; i++) { check[i] = 0; }

    cout << "\nplease enter 9 digits between 1 and 9: ";
    for(i=0; i<9; i++) {
        cin >> val;
        cout << "\n " << val;
        if (val < 1 || val > 9) { cout << "\n error " << val;
            return 0; }
        check[val-1]++;
    }
    for(i=0; i<9; i++) {
        if (check[i] != 1) { cout << "\n error with value " << i+1;
            return 0; }
    }
    cout << "\n line is valid";

    return 0;
}

A toy airplane is attached by a rope to the side of a pole, as shown from above in the figure to the right. The airplane begins flying at a constant speed and height, keeping the rope tight at all times. As a result, the rope winds around the pole, causing the airplane to fly in a spiral pattern until it crashes into the pole.

Write a program to simulate the above scenario. The input will be the airplane’s initial distance from the pole (i.e., the length of the rope) in feet, the speed of the airplane in feet per second, and the radius of the pole in feet. You may assume that the speed and radius are both greater than 0; the distance, however, may be 0 or negative. The simulation will recompute the airplane’s distance from the pole, which we define to be the length of the unwrapped portion of the rope, every millisecond (i.e., every 0.001 seconds). Each simulation step will approximate the path of the airplane as a portion of the circumference of a circle whose radius is the airplane’s distance from the pole (this is the dashed circle in the figure above). The same portion of the pole’s circumference will be wrapped by the rope; e.g., if the airplane flies for one-tenth of its circle’s circumference, then its distance from the pole will decrease by one-tenth of the pole’s circumference. The circumference of a circle is given by $2\pi r$, where $r$ is the circle’s radius. Use 3.1416 to approximate $\pi$. Note that your simulation only needs to keep track of the airplane’s distance, not its exact position. The simulation ends when the airplane’s distance becomes 0 or negative.

Your program must print the airplane’s distance initially (i.e., after 0 seconds), and after each full second of the simulation. Once the simulation ends, it should print the number of milliseconds of the simulation, from the start until the time at which the distance is less than or equal to 0.

**Example 1:**

Enter initial distance: 0.1
Enter speed: 10000
Enter pole radius: 0.1
Distance from pole after 0 seconds: 0.1
Total time: 1 milliseconds

**Example 2:**

Enter initial distance: 7.6
Enter speed: 25.7
Enter pole radius: 0.3
Distance from pole after 0 seconds: 7.6
Distance from pole after 1 seconds: 6.507012
Distance from pole after 2 seconds: 5.188726
Distance from pole after 3 seconds: 3.392076
Total time: 3749 milliseconds

**Example 3:**

Enter initial distance: -2.4
Enter speed: 33.8
Enter pole radius: 1.2
Distance from pole after 0 seconds: -2.4
Total time: 0 milliseconds
Note: All distance outputs should be accurate to 3 significant digits. For the first two tests, the total time should be exactly what is shown below; for the remaining tests, this output should be within 1 of what is shown.

Test Case 1:

Enter initial distance: 0
Enter speed: 20
Enter pole radius: 1
Distance from pole after 0 seconds: 0
Total time: 0 milliseconds

Test Case 2:

Enter initial distance: 3.1
Enter speed: 25000
Enter pole radius: 0.3
Distance from pole after 0 seconds: 3.1
Total time: 2 milliseconds

Test Case 3:

Enter initial distance: 7.4
Enter speed: 25.7
Enter pole radius: 0.2
Distance from pole after 0 seconds: 7.4
Distance from pole after 1 seconds: 6.669374
Distance from pole after 2 seconds: 5.848181
Distance from pole after 3 seconds: 4.891031
Distance from pole after 4 seconds: 3.693734
Distance from pole after 5 seconds: 1.835014
Total time: 5330 milliseconds

Retest: Do the above tests, plus the following:

Test Case 4:

Enter initial distance: -3.7
Enter speed: 24.5
Enter pole radius: 1.4
Distance from pole after 0 seconds: -3.7
Total time: 0 milliseconds

Test Case 5:

Enter initial distance: 15.6
Enter speed: 59.9
Enter pole radius: 0.7
Distance from pole after 0 seconds: 15.6
Distance from pole after 1 seconds: 12.62968
Distance from pole after 2 seconds: 8.698536
Total time: 2905 milliseconds
namespace Contest2008.SimulationAdvanced
{
    class Simulator
    {
        const float Pi = 3.1416F;

        static void Main(string[] args)
        {
            Console.Write("Enter initial distance: ");
            float distance = Single.Parse(Console.ReadLine());
            Console.Write("Enter speed: ");
            float speed = Single.Parse(Console.ReadLine());
            Console.Write("Enter pole radius: ");
            float radius = Single.Parse(Console.ReadLine());
            float eachDistance = speed / 1000;
            float postCircumference = 2 * Pi * radius;
            for (int i = 0; ; i++)
            {
                Console.WriteLine("Distance from pole after " + i + " seconds: "+
                    distance);
                for (int j = 0; j < 1000; j++)
                {
                    if (distance <= 0)
                    {
                        int time = i * 1000 + j;
                        Console.WriteLine("Total time: " + time + " milliseconds");
                        Console.ReadLine();
                        return;
                    }
                    float circumference = 2 * Pi * distance;
                    float fraction = eachDistance / circumference;
                    distance -= postCircumference * fraction;
                }
            }
        }
    }
}
Problem 2 Advanced   Matrix diagonal sums

The program reads in an integer n which is the size of the two-dimensional array. Next, the program will read \( n^2 \) integers into an n by n array by rows and prints out the sums of the two diagonals.

**Example 1**

**inputs**

Enter size of array: 3  
Enter Row: 1 2 3  
Enter Row: 4 5 6  
Enter Row: 7 8 9

**outputs**

Diagonal sums: 15 and 15

**Example 2**

**inputs**

Enter size of array: 2  
Enter Row: 1 2  
Enter Row: 2 1

**outputs**

Diagonal sums: 2 and 4
Tests 2 Advanced

Matrix diagonal sums

Formatting of inputs and outputs is not required

**First round testing**

**Test 1 - inputs**

Enter size of array: 3

Enter integers: 5 7 9 4 5 6 9 8 9

**Outputs**

Diagonal sums: 19 and 23

**Test 2 - inputs**

Enter size of array: 4

Enter integers: 1 2 3 4 4 1 2 3

3 4 1 2 2 3 4 1

**Outputs**

Diagonal sums: 4 and 12

**Second round – retesting (all of the above plus the ones below)**

**Test 1 - inputs**

Enter size of array: 3

Enter integers: 5 7 9 4 5 6 9 8 9

**Outputs**

Diagonal sums: 19 and 23

**Test 2 - inputs**

Enter size of array: 4

Enter integers: 1 2 3 4 4 1 2 3

3 4 1 2 2 3 4 1

**Outputs**

Diagonal sums: 4 and 12
Solution 2  Advanced – Array Diagonals

#include "stdafx.h"
#include <iostream>
#define MAX 25

using namespace std;
using namespace System;

int main(array<System::String ^> ^args)
{
    int d; int i,j,k; int table[MAX][MAX];
    int rightsum = 0; int leftsum = 0;
    cout <<"Enter width of array" ; cin >> d;
    for (i=0;i<d;i++)
    {
        cout <<"Enter row:"
        for (j=0;j<d;j++)
        {
            cin >> table[i][j];
        }
    }
    for (k=0;k<d;k++)
    {
        leftsum = leftsum + table[k][k];
        rightsum = rightsum + table[k][d-k-1];
    }
    cout <<"leftsum " <<leftsum;
    cout <<"rightsum " <<rightsum;
}
Write a program that reads an integer \( n \), \( 1 \leq n \leq 999,999,999 \), and prints its value in English words according to the rules illustrated in the examples below. Your input must be a single number, and your output must be consistent with the spelling, spacing, and hyphenation shown in the examples.

Example 1:

Enter \( n \): 123456789
one hundred twenty-three million four hundred fifty-six thousand seven hundred eighty-nine

Example 2:

Enter \( n \): 11012
eleven thousand twelve

Example 3:

Enter \( n \): 13000014
thirteen million fourteen

Example 4:

Enter \( n \): 15016000
fifteen million sixteen thousand

Example 5:

Enter \( n \): 217318519	
two hundred seventeen million three hundred eighteen thousand five hundred nineteen

Example 6:

Enter \( n \): 600839144	
six hundred million eight hundred thirty-nine thousand one hundred forty-four

Example 7:

Enter \( n \): 56768970	
fifty-six million seven hundred sixty-eight thousand nine hundred seventy

Example 8:

Enter \( n \): 10192300
ten million one hundred ninety-two thousand three hundred
Note: Input must be entered as a single number, and output must match exactly what is shown.

Test Case 1:

Enter n: 876543210
eight hundred seventy-six million five hundred forty-three thousand two hundred ten

Test Case 2:

Enter n: 419000502
four hundred nineteen million five hundred two

Test Case 3:

Enter n: 25001
twenty-five thousand one

Restest: Use the above tests, plus the following.

Test Case 4:

Enter n: 912390614
nine hundred twelve million three hundred ninety thousand six hundred fourteen

Test Case 5:

Enter n: 7
seven
using System;

namespace Contest2008.NumberSpellingAdvanced
{
    class Translator
    {
        /// Words to be used in the tens place. The words at
        /// indices 0 and 1 are unused, but are included to
        /// make indexing easier.
        ///
        /// private static string[] _tens = new string[]
        /// {
        ///     "", "", "twenty", "thirty", "forty", "fifty", "sixty",
        ///     "seventy", "eighty", "ninety"
        /// };
        ///
        /// Words for values 1 to 19. The word at index 0 is
        /// unused, but is included to make indexing easier.
        ///
        /// private static string[] _ones = new string[]
        /// {
        ///     "", "one", "two", "three", "four", "five", "six",
        ///     "seven", "eight", "nine", "ten", "eleven", "twelve",
        ///     "thirteen", "fourteen", "fifteen", "sixteen", "seventeen",
        ///     "eighteen", "nineteen"
        /// };
        ///
        /// Reads a positive integer containing up to 9 digits, and
        /// prints its value in English words.
        ///
        /// static void Main(string[] args)
        /// {
        ///     Console.Write("Enter n: ");
        ///     int n = Int32.Parse(Console.ReadLine());
        ///     int millions = n / 1000000;
        ///     n = n % 1000000;
        ///     if (millions > 0)
        ///     {
        ///         Console.Write(Translate3Digits(millions) + "million ");
        ///     }
        ///     int thousands = n / 1000;
        ///     n = n % 1000;
        ///     if (thousands > 0)
        ///     {
        ///         Console.Write(Translate3Digits(thousands) + "thousand ");
        ///     }
        ///     if (n > 0)
        ///     {
        ///         Console.Write(Translate3Digits(n));
        ///     }
        ///     Console.WriteLine();
        ///     Console.ReadLine();
        /// }
        ///
        /// Returns the English equivalent of n. n must be
        /// nonnegative and contain at most 3 digits. If n is
        /// 0, the empty string is returned; otherwise, the
        /// returned string will always end with a blank.
        ///
        /// private static string Translate3Digits(int n)
        /// {
        ///     string result = "";
        ///     int hundreds = n / 100;
        ///     n = n % 100;
        ///     if (hundreds > 0)
```cpp
{ 
    result += _ones[hundreds] + " hundred ";
}
if (n >= 20)
{
    int tens = n / 10;
    n = n % 10;
    result += _tens[tens];
    if (n > 0)
    {
        result += "-" + _ones[n];
    }
    result += " ";
}
else if (n > 0)
{
    result += _ones[n] + " ";
}
return result;
```
Problem 4 Advanced

Days between specified days

This program will determine the number of days between two specified days. The first input will be the day of the week of the first day of the year. This input will be one character: S (Sunday), M (Monday), T (Tuesday), W (Wednesday), U or R (Thursday), F (Friday), or A (Saturday). The next input will be a specific day- the month, the day of the week and ordinal of the day. E.g. 5, F, 4 will be the fourth Friday of the month of May. The next input will be another specific day. The output will be the number of days between those two days, not including either day. Assume that all dates are in the same year and that the first specified day occurs before the second specified day. Assume that all dates are in the month specified.

Example 1:

Enter first day of the year: M
Enter month, day of the week, ordinal: 2 F 1
Enter month, day of the week, ordinal: 2 F 3
Output: 13 days

Example 2:

Enter first day of the year: T
Enter month, day of the week, ordinal: 1 W 3
Enter month, day of the week, ordinal: 3 F 4
Output: 64 days
Problem 4 Advanced

Days between specified days

First round of testing

Test 1:
Enter first day of the year: F
Enter month, day of the week, ordinal: 1 W 2
Enter month, day of the week, ordinal: 5 W 2
Outpht: 118 days

Test 2:
Enter first day of the year: T
Enter month, day of the week, ordinal: 1 W 3
Enter month, day of the week, ordinal: 3 F 4
Outpht: 64 days

Second round of testing – the above tests plus the ones below

Test 3:
Enter first day of the year: U
Enter month, day of the week, ordinal: 7 W 1
Enter month, day of the week, ordinal: 8 A 1
Outpht: 30 days

Test 4:
Enter first day of the year: F
Enter month, day of the week, ordinal: 3 T 5
Enter month, day of the week, ordinal: 6 T 5
Outpht: 90 days
Solution 4 Advanced – Calendar

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

using namespace std;
using namespace System;

int dayoftheweek(int firstday, int day) {
    // returns numerical day of the week for a day, given the first day of the year.
    // assumes non-leap year
    int temp;
    temp = day%7+firstday-1;
    if (temp > 7) temp = temp -7;
    return temp;
};

int dayoftheyear(int month) {
    //returns numerical day of the year given the numerical month
    //assumes non-leap year
    if (month == 1) return 1;
    if (month == 2) return 32;
    if (month == 3) return 60;
    if (month == 4) return 91;
    if (month == 5) return 121;
    if (month == 6) return 152;
    if (month == 7) return 182;
    if (month == 8) return 213;
    if (month == 9) return 244;
    if (month == 10) return 274;
    if (month == 11) return 305;
    if (month == 12) return 335;
    return 0;
};

int doy(int month, int firstday, int dow, int ordinal){
    int doyformonth,temp,doyforday,maxdoy;
    doyformonth = dayoftheyear(month);
    if(month == 12) {maxdoy = 366;}
    else {maxdoy = dayoftheyear(month+1);}
    temp = dow - dayoftheweek(firstday,doyformonth);
    if (temp <0) temp = temp+7;
    doyforday = temp + doyformonth + (ordinal-1)*7;
    if (doyforday >= maxdoy) {cout<<"\n Not in same month";
    return 0;}
    return doyforday;
};

int daytonum(char day){
    int num;
    if (day == 's' || day == 'S') { num = 1;}
    if (day == 'm' || day == 'M') { num = 2;}
    if (day == 't' || day == 'T') { num = 3;}
```
if (day == 'w' || day == 'W') { num = 4;
if (day == 'r' || day == 'R') { num = 5;
if (day == 'f' || day == 'F') { num = 6;
if (day == 'a' || day == 'A') { num = 7;
return num;
}

int main(array<System::String ^ > ^args) {
    int month1, month2, num1, num2, dyl, dy2, dow1, dow2, fnum;
    char firstday, day1, day2, q;
    cout << "\n enter the day of the week of the first day of the year: ";
cin >> firstday;
cout << "\n enter the number of the first month: "; cin >> month1;
cout << "\n enter the char for the first day: "; cin >> day1;
cout << "\n enter the ordinal of the first day: "; cin >> num1;
cout << "\n enter the number of the second month: "; cin >> month2;
cout << "\n enter the char for the second day: "; cin >> day2;
cout << "\n enter the ordinal of the second day: "; cin >> num2;

dow1 = daytonum(day1); dow2 = daytonum(day2);
fnum = daytonum(firstday);
dy1 = doy(month1, fnum, dow1, num1); cout << "\ndy1 " <<dy1;
dy2 = doy(month2, fnum, dow2, num2); cout << " dy2 " <<dy2;

cout << "\n the number of day is " <<dy2 - dyl - 1;
return 0;
}
A line in the $x/y$ plane can be described by an equation of the form

$$y = mx + b$$

(1)

where $m$ denotes the slope, and $b$ denotes the $y$-intercept. Write a program that reads the slope and $y$-intercept of three lines and outputs the area of the triangle bounded by the lines. The area of a triangle is given by $bh/2$, where $b$ is the length of one of its sides (the base), and $h$ (the height) is the perpendicular distance from the base to the vertex opposite the base. You may assume that all slopes are distinct, but you may make no assumptions about the order in which the lines are entered. Note that if all three lines intersect at the same point, the area of the triangle is 0.

Hints: The $x$-coordinate of the intersection point of two lines with different slopes is that value of $x$ that makes the two $y$ values equal. Given an $x$-coordinate, the corresponding $y$-coordinate on a given line can be obtained by applying Equation (1) above. The area of the triangle can be easily computed by breaking it into two triangles with a vertical line as shown in the figure below. These triangles share a common base, whose length is the vertical distance between the intersection point with median $x$-value and the line opposite that point. The heights are the horizontal distances between each of the other two intersection points and the vertical line.

Example 1:

Enter slope 0: -1
Enter y-intercept 0: 0
Enter slope 1: 0.75
Enter y-intercept 1: 0
Enter slope 2: 0
Enter y-intercept 2: 3
Area = 10.5

Example 2:

Enter slope 0: 1.2
Enter y-intercept 0: -2.3
Enter slope 1: -2.1
Enter y-intercept 1: 4.5
Enter slope 2: -0.3
Enter y-intercept 2: 2.7
Area = 2.227273
5 Advanced — Area Between Lines
Test Cases

Note: Results should be accurate to 3 significant digits.

Test Case 1:

Enter slope 0: -1.2
Enter y-intercept 0: 3.4
Enter slope 1: 5.6
Enter y-intercept 1: -7.8
Enter slope 2: -9
Enter y-intercept 2: 1.2
Area = 6.759813

Test Case 2:

Enter slope 0: 5.6
Enter y-intercept 0: -7.8
Enter slope 1: -9
Enter y-intercept 1: 1.2
Enter slope 2: -1.2
Enter y-intercept 2: 3.4
Area = 6.759813

Test Case 3:

Enter slope 0: -9
Enter y-intercept 0: 1.2
Enter slope 1: -1.2
Enter y-intercept 1: 3.4
Enter slope 2: 5.6
Enter y-intercept 2: -7.8
Area = 6.759813

Retest: Do the above tests, plus the following:

Test Case 4:

Enter slope 0: 1
Enter y-intercept 0: 2
Enter slope 1: 3
Enter y-intercept 1: 4
Enter slope 2: 5
Enter y-intercept 2: 6
Area = 0

Test Case 5:

Enter slope 0: 9.8
Enter y-intercept 0: -7.6
Enter slope 1: -5.4
Enter y-intercept 1: 3.2
Enter slope 2: 1
Enter y-intercept 2: -9.8
Area = 9.641283
namespace Contest2008.AreaAdvanced
{
    using System;

    class LineAndPoint
    {
        public float Slope;
        public float YIntercept;
        public float X;
        public float Y;
    }

    class Area
    {
        static void Main(string[] args)
        {
            LineAndPoint[] lp = new LineAndPoint[3];
            for (int i = 0; i < 3; i++)
            {
                lp[i] = new LineAndPoint();
                Console.Write("Enter slope " + i + ": ");
                lp[i].Slope = Single.Parse(Console.ReadLine());
                Console.Write("Enter y-intercept " + i + ": ");
                lp[i].YIntercept = Single.Parse(Console.ReadLine());
            }

            // We'll store an intersection point with the line that
            // doesn't contain it.
            FindIntersection(lp[0], lp[1], lp[2]);
            FindIntersection(lp[0], lp[2], lp[1]);
            FindIntersection(lp[1], lp[2], lp[0]);

            // Sort the points by x-coordinate.
            if (lp[1].X < lp[0].X)
            {
                Swap(lp, 1, 0);
            }
            if (lp[2].X < lp[1].X)
            {
                Swap(lp, 2, 1);
                if (lp[1].X < lp[0].X)
                {
                    Swap(lp, 1, 0);
                }
            }
            // The base is the vertical distance between the point and line stored in
            // lp[1].

            // The sum of the heights is the horizontal distance between the points in
            // lp[0] and lp[2].
            float height = lp[2].X - lp[0].X;

            // The area is half the base times the sum of the heights.
            float area = baseLen * height / 2;
            Console.WriteLine("Area = " + area);
            Console.ReadLine();
        }
    }

    static void Swap<T>(T[] list, int a, int b)
    {
        T temp = list[a];
        list[a] = list[b];
        list[b] = temp;
    }

    static void FindIntersection(LineAndPoint a, LineAndPoint b, LineAndPoint c)
    {
        if (float.IsInfinity(x) || float.IsInfinity(y) || float.IsInfinity(z))
        {
            // Intersection is not possible.
            return;
        }
        float t = x / z;
        float s = y / z;
        if (0 <= t && t <= 1 && 0 <= s && s <= 1)
        {
            x = a.X + t * (b.X - a.X);
            y = a.Y + t * (b.Y - a.Y);
            Console.WriteLine("Intersection at (" + x + ", " + y + ")");
        }
    }
}
/// Stores as the point in intersection the intersection of the lines in line1
/// and line2.
///
/// static void FindIntersection(LineAndPoint line1, LineAndPoint line2,
/// LineAndPoint intersection)
/// {
/// intersection.X =
/// (line1.YIntercept - line2.YIntercept)/(line2.Slope - line1.Slope);
/// intersection.Y = line1.Slope * intersection.X + line1.YIntercept;
/// }

/// Swaps the elements in array given by the two indices.
///
/// static void Swap(LineAndPoint[] array, int index1, int index2)
/// {
/// LineAndPoint t = array[index1];
/// array[index1] = array[index2];
/// array[index2] = t;
/// }

Problem 6 Advanced

Sudoku is a puzzle that uses a 9 by 9 array. The game requires that each row, each column, each 3by3 sub-array contain one and exactly one of the integers 1 through 9.

The program you will write will take a 4 by 4 array that contains 4 sub-array of size 2 by 2. Each row, each column, and each of the four 2 by 2 sub-arrays must contain exactly one value of each of the integers from 1 through 4. Read the array, one row at a time.

**Correct:**

Row1: 1 2 3 4  
Row2: 4 3 2 1  
Row3: 3 4 1 2  
Row4: 2 1 4 3

Row1: 1 2 3 4  
Row2: 4 3 1 2  
Row3: 2 3 4 1  
Row4: 4 1 2 3

Row1: 1 2 3 4  
Row2: 4 3 1 2  
Row3: 2 1 3 4  
Row4: 3 4 2 1

**Incorrect:**

Row1: 1 2 3 4  
Row2: 2 3 4 1  
Row3: 3 4 1 2  
Row4: 4 1 2 3 (each subarray is incorrect)

Row1: 1 2 3 1  
Row2: 4 3 4 4  
Row3: 3 4 1 2  
Row4: 2 1 2 3 (row 1,2,4 wrong)

Row1: 1 2 3 4  
Row2: 2 3 4 1  
Row3: 2 1 3 4  
Row4: 3 4 2 1 (column 1,4 and row1,4 wrong)

Row1: 1 2 3 2

**Example 1:**

Enter row1: 1 2 3 4  
Enter row2: 2 3 4 1  
Enter row3: 3 4 1 2  
Enter row4: 4 1 2 3

Sudoku is not correct

**Example 2:**

Enter row1: 1 2 3 4  
Enter row2: 4 3 2 1  
Enter row3: 3 4 1 2  
Enter row4: 2 1 4 3
Sudoku is correct
Solution 6 Advanced – 4x4 Sudoku

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

using namespace std;
using namespace System;

int main(array<System::String ^> ^args)
{
    int sudoku[4][4]; int row, col; int check[4]; int val,i;

    cout << "\n enter 4by4 sudoku by rows: \n";
    for(col = 0; col < 4; col++)
    {
        cout << "\nEnter row:"
        for (row = 0; row < 4; row++)
        {
            cin >> val;
            if (val < 1 || val > 4) {cout << "\nerror " << val ;
            cin >> q; return 0;}
            sudoku[col][row] = val;
        }
    }

    //check each row
    for(row = 0; row < 4; row++)
    {
        for (i = 0; i < 4; i++)
        {
            check[i] = 0;
        }
        for (col = 0; col < 4; col++)
        {
            check[sudoku[col][row]-1]++;
        }
        for (i = 0; i<4; i++)
        {
            if (check[i] != 1) {cout<<"\n row " <<row+1<<" in error ";
            return 0;
        }
    }

    //check each col
    for(col = 0; col < 4; col++)
    {
        for (i = 0; i < 4; i++)
        {
            check[i] = 0;
        }
        for (row = 0; row < 4; row++)
        {
            check[sudoku[col][row]-1]++;
        }
        for (i = 0; i<4; i++)
        {
            if (check[i] != 1) {cout<<"\n col " <<col+1<<" in error ";
            return 0;
        }
    }
    cout << "\n col " <<col+1<<" is correct.";
}

    //check each square
    for(col = 0; col < 4; col=col+2){
for(row = 0; row < 4; row=row+2){
    for (i = 0; i < 4; i++){
        check[i] = 0;
    }
    check[sudoku[col][row-1]]++;
    check[sudoku[col][row+1]-1]++;
    check[sudoku[col+1][row]-1]++;
    check[sudoku[col+1][row+1]-1]++;
    for (i = 0; i<4; i++) {
        if (check[i] != 1) {cout<<"\n square "
            <<col+1<<row+1<<" in error "; return 0;
        }
    }
    cout << "\n square "<<col+1 <<row+1 <<" is correct";
}
    cout<<"\nsudoku is correct!";
cout<<"\nenter q;"; cin>>q;
return 0;
}