B1 – sprinkler coverage

Problem Statement

There will be two irrigation sprinklers located at opposite corners of a field. Assume that each sprinkler covers a quarter of a circle. Assume that the field is long enough that the sprinklers can cover the full width without overlapping. Given the length and the width of the field, determine the maximum percentage of the field that can be covered with these two sprinklers. The total area of a circle is 3.14159 * radius * radius. Show at least one place to the right of the decimal.

Examples

length 3	width 10	width 10.5
width 1	length 20	length 17.9
max % 52.3	max % 78.5	max % 92.14

2 Beginning — Recurrences

A *second-order linear homogeneous recurrence* is a function f that satisfies an equation of the form

$$f(n) = af(n-2) + bf(n-1)$$

whenever $n \geq 2$. For example, if

$$\begin{aligned} a &= -3 \\ b &= 4 \\ f(0) &= 1 \\ f(1) &= 2 \end{aligned}$$

then

$$\begin{aligned} f(2) &= -3f(0) + 4f(1) = -3 + 8 = 5 \\ f(3) &= -3f(1) + 4f(2) = -6 + 20 = 14 \\ f(4) &= -3f(2) + 4f(3) = -15 + 56 = 41 \end{aligned}$$

etc.

Write a program that takes as input integer values for a, b, $f(0)$, $f(1)$, and n, and outputs the value of $f(n)$. You may assume that $0 \leq n \leq 20$.

Example 1:

```
Enter a: -3
Enter b: 4
Enter f(0): 1
Enter f(1): 2
Enter n: 4

f(n) = 41
```

Example 2:

```
Enter a: 2
Enter b: -3
Enter f(0): 0
Enter f(1): 2
Enter n: 0

f(n) = 0
```

B3 Tic-tac-toe

Problem Statement

The spaces on a tic-tac-toe board can be represented by the numbers in a magic square. The advantage is that the sum of any row, column, or diagonal is the same. So, any winning combination has values that sum to the same number, in this case, 15.

Given four "X" moves, determine if any three are a completed row, column, or diagonal.

6	7	2
1	5	9
8	3	4

Examples:

X1: 6
X2: 2
X3: 4
X4: 8
Output: no win

X1: 6
X2: 9
X3: 4
X4: 5
Output: Win

X1: 1
X2: 9
X3: 4
X4: 5
Output: Win

4 Beginning — Particles

Two particles are traveling directly toward each other. We wish to determine how soon they will collide and how far each particle will travel before the collision. Write a program that takes as input the speeds of the two particles and their distance apart (all as floating-point numbers) and outputs the desired information. You may assume that all inputs are positive.

Example 1:

```
Enter speed of first particle: 10
Enter speed of second particle: 20
Enter distance: 60

The particles collide after 2 time units.
The first particle traveled a distance of 20.
The second particle traveled a distance of 40.
```

Example 2:

```
Enter speed of first particle: 12.3
Enter speed of second particle: 98.7
Enter distance: 55.5

The particles collide after 0.5 time units.
The first particle traveled a distance of 6.15.
The second particle traveled a distance of 49.35.
```

B5 Driving Times

Problem Statement

There are two possible routes from point A to point D. On the first route, there are 30 miles of City driving and then 90 miles of Inter-state driving. On the second route, there are 25 miles of City driving and then 80 miles of Highway (non-interstate) driving.

Input the average speeds, in miles per hour, for City driving, Highway driving, and Inter-state driving. Determine which route takes the least minutes and print the two times.

Examples

City: 30	City: 25	City: 25.5
High: 40	High: 60	High: 55
Inter: 45	Inter: 70	Inter: 75.5
First route is faster	Second route is faster	First route is faster
180 min; 170 min	149.1 min; 140 min	142.111 min; 154.824 min

6 Beginning — Football Scoring

We wish to determine the number of ways of achieving a given score in football. For the purposes of this program, we assume that there are only two ways of scoring:

- A field goal: 3 points
- A touchdown: 7 points

For example, there are 3 ways to score 45 points:

- 15 field goals
- 8 field goals and 3 touchdowns
- 1 field goal and 6 touchdowns

Your program should take as input a single nonnegative integer score no greater than 100, and it should output the number of ways of achieving that score.

Example 1:

```
Enter score: 45

Number of ways: 3
```

Example 2:

```
Enter score: 0

Number of ways: 1
```

```
/* 2 Beginning - Recurrences
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Recurrences.Beginning
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter a: ");
            int cPrev = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter b: ");
            int cLast = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter f(0): ");
            int prev = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter f(1): ");
            int last = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter n: ");
            int n = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine();
            if (n == 0)
            {
                Console.WriteLine(prev);
            }
            else
            {
                for (int i = 1; i < n; i++)
                {
                    int next = cPrev * prev + cLast * last;
                    prev = last;
                    last = next;
                }
                Console.WriteLine("f(n) = " + last);
            }
            Console.ReadLine();
        }
    }
}
```

B3 Tic-tac-toe Solution

```
int x1, x2, x3, x4, t, i;


std::cout << "\nEnter first X move: ";
std::cin >> x1;
std::cout << "\nEnter second X move: ";
std::cin >> x2;
std::cout << "\nEnter third X move: ";
std::cin >> x3;
std::cout << "\nEnter fourth x move: ";
std::cin >> x4;
t = x1 + x2 + x3 + x4;
if (t < 16) {
	std::cout << "\nNo win - too small";
}
if (t > 25) {
	std::cout << "\nNo win - too large";
}
if (t >= 16 && t <= 25) {
	i = t - 15;
	if (x1 == i || x2 == i || x3 == i || x4 == i) {
		std::cout << "\nWin";
	}
	else { std::cout << "\nNo Win"; }
}
```

```
/* 4 Beginning - Particles
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Particles.Beginning
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter speed of first particle: ");
            float v1 = Convert.ToSingle(Console.ReadLine());
            Console.Write("Enter speed of second particle: ");
            float v2 = Convert.ToSingle(Console.ReadLine());
            Console.Write("Enter distance: ");
            float d = Convert.ToSingle(Console.ReadLine());

            Console.WriteLine();
            float t = d / (v1 + v2);
            Console.WriteLine("The particles collide after " + t + " time units.");
            Console.WriteLine("The first particle traveled a distance of " + (v1 * t)
                + ".");
            Console.WriteLine("The second particle traveled a distance of " + (v2 * t)
                + ".");
            Console.ReadLine();
        }
    }
}
```

B5 Driving

Solution

```
float  distAB, distBD, distAC, distCD;
float aveCity, aveInter, aveHigh;
float timeABD, timeACD;

distAB = 30; // city travel
distBD = 90; // Inter travel
distAC = 25; // city travel
distCD = 80; // highway travel

std::cout << "\nEnter average speed in MPH for city streets: ";
std::cin >> aveCity;
std::cout << "\nEnter average speed in MPH for Interstates: ";
std::cin >> aveInter;
std::cout << "\nEnter average speed in MPH for highways: ";
std::cin >> aveHigh;

timeABD = 60 * (distAB / aveCity + distBD / aveInter);
timeACD = 60 * (distAC / aveCity + distCD / aveHigh);
std::cout << "\n" << timeABD << " " << timeACD;
if (timeABD <= timeACD)
	{
		std::cout << "\n the first route is faster.  It will take "
		<< timeABD << " minutes";
	}
	else {
		std::cout << "\n the second route is faster.  It will take "
		<< timeACD << " minutes";
	}
```

```
/* 6 Beginning - Football Scoring
 */
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace FootballScoring.Beginning
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter score: ");
            int score = Convert.ToInt32(Console.ReadLine());

            // The number of combinations can be computed with the following formula:
            //
            // int n = (score + 21 - (score % 3) * 7) / 21;
            //
            // However, the following code is easier to understand.

            int n = 0;
            for (int i = 0; i <= score; i += 7)
            {
                if ((score - i) % 3 == 0)
                    // or if (((score - i) / 3) * 3 == score - i)
                {
                    n++;
                }
            }
            Console.WriteLine();
            Console.WriteLine("Number of ways: " + n);
            Console.ReadLine();
        }
    }
}
```