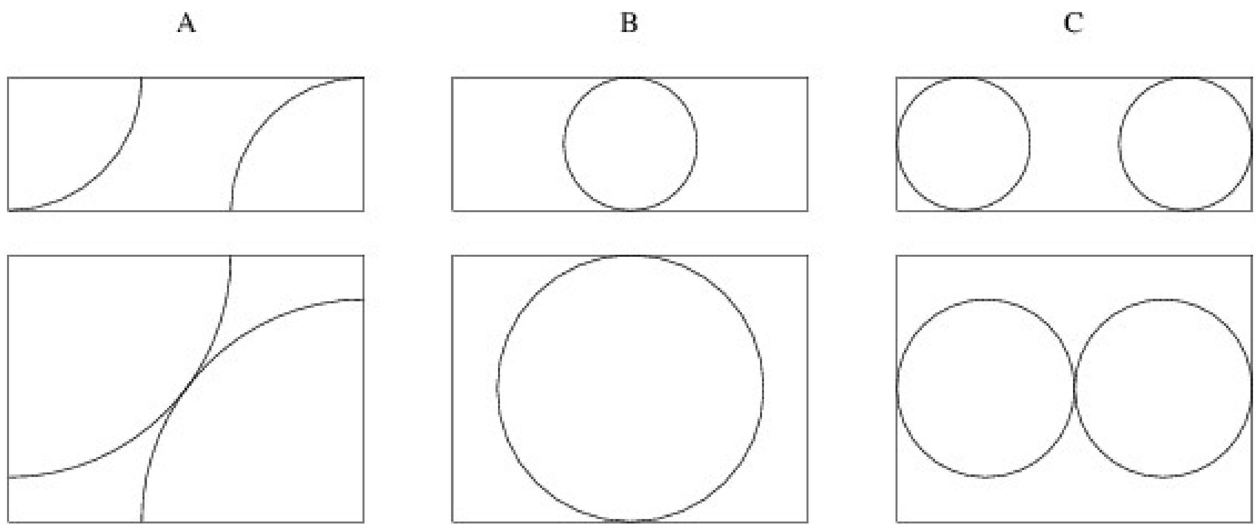
A1 – sprinkler coverage

Problem Statement

There are three possible configurations of irrigation sprinklers: A) two quarter systems installed at opposite corners, B) one full circle system install in the center of the field, or C) two same-sized, full circle systems installed at radius-length from each end of the field (see figures below, which show the configurations for fields of different widths). Make sure that multiple systems do not overlap, as shown in the lower figures below. Given the length and the width of the field, determine which system covers the maximum percentage of the field. The total area of a circle is 3.14159 * radius * radius. Note that for configuration A, the radius is half the diagonal. The length of the diagonal is the square root of (length * length + width * width). (Note that you don't necessarily need to compute the square root, as it will disappear when you square the radius.) Output the system type and percentage of coverage for each option. You may assume that the width is no greater than the length.



Examples

W 10.5
L 20.6
A best
A: 80.06
B: 40.03
C: 77.04

W 10
L 10
A best
A: 78.53
B: 78.53
C: 39.27

Note: Choosing B as best is also acceptable.

W 10
L 17
A best
A: 89.85
B: 46.19
C: 66.76

2 Advanced — Recurrences

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

$$f(n) = a_0 f(n-1) + a_1 f(n-2) + \cdots + a_{k-1} f(n-k)$$

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

$$\begin{aligned} k &= 2 \\ a_0 &= 4 \\ a_1 &= -3 \\ f(0) &= 1 \\ f(1) &= 2 \end{aligned}$$

then

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

etc.

Write a program that takes as input integer values for k, $a_0, \ldots, a_{k-1}$, $f(0), \ldots, f(k-1)$, and n, and outputs the value of $f(n)$. You may assume that $1 \leq k \leq 5$ and $0 \leq n \leq 20$.

Example 1:

```
Enter k: 2
Enter a0: 4
Enter a1: -3
Enter f(0): 1
Enter f(1): 2
Enter n: 4

f(n) = 41
```

Example 2:

```
Enter k: 5
Enter a0: 1
Enter a1: 2
Enter a2: 3
Enter a3: 4
Enter a4: 5
Enter f(0): 0
Enter f(1): 1
Enter f(2): 2
Enter f(3): 3
Enter f(4): 4
Enter n: 0

f(n) = 0
```

A3 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 same number, in this case 15.

Given three “X” moves and three “O” moves, determine if “X” or “O” has already won. If not, does “X” have a winning move or if not, does “X” have a blocking move.

6	7	2
1	5	9
8	3	4

Examples:

X1: 6	X1: 6	X1: 6
O1: 9	O1: 9	O1: 8
X2: 1	X2: 2	X2: 2
O2: 7	O2: 5	O2: 4
X3: 5	X3: 7	X3: 9
O3: 8	O3: 1	O3: 7
Output: Play 4 to win	Output: X won	Output: Play 3 to block

4 Advanced — Particles

A particle is trapped in a box whose size is $1000 \times 1000 \times 1000$ in some unit of length. Write a program that takes as input the particle's initial position and velocity and a time duration, and determines the particle's position after the given time duration has elapsed. The position will be given as x, y, and z coordinates, all nonnegative floating-point numbers no more than 1000. The velocity will be given as three floating-point numbers at least -100 and no greater than 100. These three numbers give the change in each coordinate over one time unit, assuming the particle does not hit one of the walls of the box. The time duration will be a nonnegative integer no greater than 100.

You are to compute the particle's final position by simulating its movement, one time unit at a time. For example, suppose its initial position is $(850, 100, 525)$, and its initial velocity is $(100, -75, 30)$. Then after one time unit, its position is $(950, 25, 555)$. After two time units, if there were no walls, its new position would be $(1050, -50, 585)$. To account for the walls, we need to do the following:

- Change the velocity by negating each dimension in which a wall was struck. Thus, the new velocity will be $(-100, 75, 30)$.
- Change the position by assuming the particle reversed its direction in any dimension in which a wall was struck, at the point in time when that wall was struck. Thus, the final 50 distance units in both the x and y dimensions will be traveled in the opposite direction. The new position is therefore $(950, 50, 585)$.

After a third time unit, the new position is $(850, 125, 615)$.

Example 1:

```
Enter initial x-coordinate: 850
Enter initial x-velocity: 100
Enter initial y-coordinate: 100
Enter initial y-velocity: -75
Enter initial z-coordinate: 525
Enter initial z-velocity: 30
Enter time duration: 3

Final position: (850, 125, 615).
```

Example 2:

```
Enter initial x-coordinate: 0
Enter initial x-velocity: -100
Enter initial y-coordinate: 1000
Enter initial y-velocity: 0
Enter initial z-coordinate: 12.34
Enter initial z-velocity: 99.999
Enter time duration: 100

Final position: (0, 1000, 12.24086).
```

A5 Driving Times

Problem Statement

There are two possible routes from point A to point D. Each route might have up to 10 segments. Each segment is either city streets, non-interstate highways, or Interstate roads. Input the number of segments, then prompt for each length in miles and type of road (C, H, or I).

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.

Then allow the user to change one of the speeds to see the effect on the times and on which route is faster.

Example

```
Route 1 number: 3
dist: 70 type: I
dist: 55 type: H
dist: 30 type: C
Route 2 number: 2
dist: 70 type: I
dist: 60 type: C
ave speed City: 15
ave speed High: 55
ave speed Inter: 70
output:
First is faster;
240 min; 300 min
Change H 65
First is faster;
230.76min; 300 min
```

```
Route 1 number: 4
dist: 100 type: I
dist: 100 type: H
dist: 50 type: C
dist: 100 type: H
Route 2 number: 3
dist: 50 type: H
dist: 200 type: I
dist: 70 type: C
ave speed City: 30
ave speed High: 65
ave speed Inter: 70
output:
Second is faster;
364.615 min; 346.154 min
Change H 55
First is faster;
398.182 min; 354.545 min
```

```
Route 1 number: 2
dist: 65.7 type: H
dist: 41.2 type: C
Route 2 number: 1
dist: 140.8 type: I
ave speed City: 20.6
ave speed High: 65
ave speed Inter: 70.4
output:
second is faster;
180.646 min; 120 min
Change C 41.2
second is faster;
120.646 min; 120 min
```

6 Advanced — 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 five ways of scoring:

- A safety: 2 points
- A field goal: 3 points
- A touchdown with no extra points: 6 points
- A touchdown with one extra point: 7 points
- A touchdown with two extra points: 8 points

For example, there are 4 ways to score 9 points:

- 3 safeties and 1 field goal
- 1 safety and 1 touchdown with one extra point
- 1 field goal and 1 touchdown with no extra point
- 3 field goals

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: 9

Number of ways: 4
```

Example 2:

```
Enter score: 0

Number of ways: 1
```

A1 Sprinklers Solution

```
double a, p2Q, pC, p2C, r, s, l, w, t;


std::cout << "\nenter w: ";
std::cin >> w;
std::cout << "\nenter l: ";
std::cin >> l;
a = w*l;
if (w > l) { t = l; l = w; w = t; }
// two quarter
if (w > .5*(sqrt(w*w + l*l))) {
       r = .5*(sqrt(w*w + l*l));
}
else { r = w; }
s = 2 * 0.25*3.14159*r*r;
p2Q = s / a;
std::cout << "\n " << p2Q << "%";

// circle
r = w / 2;
s = 3.14159* r * r;
pC = s / a;

// two circle
if (2*w > l) { r = l / 4; }
else { r = w / 2; }
s = 2 * 3.14159* r *r ;
p2C = s / a;
std::cout << "\nA: " << 100*p2Q << " B: " << 100*pC << " C: " << 100*p2C;
std::cout << "\nquit? enter q: ";
std::cin >> i;
```

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

namespace Recurrences.Advanced
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] coefficients = new int[21];
            int[] values = new int[21];
            Console.Write("Enter k: ");
            int order = Convert.ToInt32(Console.ReadLine());
            for (int i = 0; i < order; i++)
            {
                Console.Write("Enter a" + i + ": ");
                coefficients[i] = Convert.ToInt32(Console.ReadLine());
            }
            for (int i = 0; i < order; i++)
            {
                Console.Write("Enter f(" + i + "): ");
                values[i] = Convert.ToInt32(Console.ReadLine());
            }
            Console.Write("Enter n: ");
            int n = Convert.ToInt32(Console.ReadLine());
            for (int i = order; i <= n; i++)
            {
                int sum = 0;
                for (int j = 0; j < order; j++)
                {
                    sum += coefficients[j] * values[i - j - 1];
                }
                values[i] = sum;
            }
            Console.WriteLine();
            Console.WriteLine("f(n) = " + values[n]);
            Console.ReadLine();
        }
    }
}
```

A3 Tic Tac Toe Solution

```
int x1, x2, x3, x4, o1, o2, o3, o4, x, o, t, i, b, done;

std::cout << "\nEnter first X move: ";
std::cin >> x1;
std::cout << "\nEnter first O move: ";
std::cin >> o1;
std::cout << "\nEnter second X move: ";
std::cin >> x2;
std::cout << "\nEnter second O move: ";
std::cin >> o2;
std::cout << "\nEnter third X move: ";
std::cin >> x3;
std::cout << "\nEnter third O move: ";
std::cin >> o3;
done = 0;
if (x1 + x2 + x3 == 15) {
        std::cout << "\nX has already won";
        }
else {
        if (o1 + o2 + 03 == 15) {
                std::cout << "\nO has already won";
        }
        else {
                b = 0;
                t = 15 - (x1 + x2); if (t == o1 || t == o2 || t == o3) {
                std::cout << "\nBlocked 1"; b++;
        }
        else {
                std::cout << "\nPlay " << t;
        }
                t = 15 - (x1 + x3); if (t == o1 || t == o2 || t == o3) {
                        std::cout << "\nBlocked 2"; b++;
                }
                else {
                        std::cout << "\nPlay " << t;
                }
                t = 15 - (x2 + x3); if (t == o1 || t == o2 || t == o3) {
                        std::cout << "\nBlocked 3"; b++;
                        }
                else {
                        std::cout << "\nPlay " << t;
                }
                if (b == 3) {
                        "\nAll wins blocked";
                        b = 0;
                        t = 15 - (o1 + o2);
                        if (t == x1 || t == x2 || t == x3) {
                                std::cout << "\nBlocked 4"; b++;
                                }
                        else {
                                std::cout << "\nblock with " << t;
                        }
                        t = 15 - (o1 + o3); if (t == x1 || t == x2 || t == x3) {
                                std::cout << "\nBlocked 5"; b++;
                                }
                        else {
                                std::cout << "\nblock with " << t;
                        }
                        t = 15 - (o2 + o3); if (t == x1 || t == x2 || t == x3) {
                                std::cout << "\nBlocked 5"; b++;
                        }
                        else {
                                std::cout << "\nblock with " << t;
                        }
                        if (b == 3) {
                                std::cout << "\nNo good plays left";
                        }
                }
        }
```

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

namespace Particles.Advanced
{
    class Program
    {
        static void Main(string[] args)
        {
            float[] pos = new float[3];
            float[] vel = new float[3];
            for (int i = 0; i < 3; i++)
            {
                Console.Write("Enter initial " + (char)('x' + i) + "-coordinate: ");
                pos[i] = Convert.ToSingle(Console.ReadLine());
                Console.Write("Enter initial " + (char)('x' + i) + "-velocity: ");
                vel[i] = Convert.ToSingle(Console.ReadLine());
            }
            Console.Write("Enter time duration: ");
            int t = Convert.ToInt32(Console.ReadLine());
            for (int i = 0; i < t; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    pos[j] += vel[j];
                    if (pos[j] > 1000)
                    {
                        vel[j] = -vel[j];
                        pos[j] = 2000 - pos[j];
                    }
                    else if (pos[j] < 0)
                    {
                        vel[j] = -vel[j];
                        pos[j] = -pos[j];
                    }
                }
            }
            Console.WriteLine();
            Console.WriteLine("Final position: (" + pos[0] + ", " + pos[1] + ", "
                + pos[2] + ").");
            Console.ReadLine();
        }
    }
}
```

A5 Driving

Solution

```
        int i;
        float aveCity, aveInter, aveHigh, newspeed, ntimepath1, ntimepath2;
        float timepath1, timepath2, newtime1, newtime2;
        char t, newtype;
        float path1[10], path2[10];
        char path1type[10], path2type[10];
        int length1, length2;

        std::cout << "\nEnter number of steps on path1: "; std::cin >> length1;
        for (i = 0; i < length1; i++) {std::cout << "\nEnter miles on step " << i+1 <<
                ":  "; std::cin >> path1[i];
                std::cout << "\nEnter type of road for step " << i+1 <<": ";
                std::cin >> path1type[i];}
        std::cout << "\nEnter number of steps on path2: "; std::cin >> length2;
        for (i = 0; i < length2; i++) {std::cout << "\nEnter miles on step " << i+1 <<
                 " :  "; std::cin >> path2[i];
                std::cout << "\nEnter type of road for step " << i+1 << " : ";
                std::cin >> path2type[i];
                }
        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;

        timepath1 = 0;
        for (i = 0; i < length1; i++)
                {
                if (path1type[i] == 'C') {timepath1 = timepath1 + path1[i] / aveCity;
                              std::cout << "\nC " << timepath1;}
                if (path1type[i] == 'I') {timepath1 = timepath1 + path1[i] / aveInter;
                              std::cout << "\nI " << timepath1;}
                if (path1type[i] == 'H') {timepath1 = timepath1 + path1[i] / aveHigh;
                              std::cout << "\nH " << timepath1;} }
                timepath1 = 60 * timepath1;
        timepath2 = 0;
        for (i = 0; i < length2; i++)
                {
                if (path2type[i] == 'C') {timepath2 = timepath2 + path2[i] / aveCity;
                              std::cout << "\nC2 " << timepath2; }
                if (path2type[i] == 'I') {timepath2 = timepath2 + path2[i] / aveInter;
                              std::cout << "\nI2 " << timepath2; }
                if (path2type[i] == 'H') {  timepath2 = timepath2 + path2[i] / aveHigh;
                              std::cout << "\nH2 " << timepath2; }
                }
        timepath2 = 60 * timepath2; std::cout << "\n" << timepath1<< " " << timepath2;
        if (timepath1 < timepath2)
                {std::cout << "\n the first route is faster.  It will take "
                << timepath1 << " minutes"; }
                else { std::cout << "\n the second route is faster.  It will take "
                << timepath2 << " minutes"; }
                t = 'n';
                std::cout << "\nWould you like to try a different speed?: "; std::cin >> t;
```

```
if (t == 'Y' || t == 'y') {
        std::cout << "\nWhich speed type?: "; std::cin >> newtype;
        std::cout << "\nwhat is new speed?: "; std::cin >> newspeed;
        if (newtype == 'C') { aveCity = newspeed; }
        if (newtype == 'I') { aveInter = newspeed; }
        if (newtype == 'H') { aveHigh = newspeed;  }

        ntimepath1 = 0;
        for (i = 0; i < length1; i++)
                {
                if (path1type[i] == 'C') {
                        ntimepath1 = ntimepath1 + path1[i] / aveCity;
                        std::cout << "\nC3 " << ntimepath1;}
                if (path1type[i] == 'I') {
                        ntimepath1 = ntimepath1 + path1[i] / aveInter;
                        std::cout << "\nI3 " << ntimepath1;}
                if (path1type[i] == 'H') {
                        ntimepath1 = ntimepath1 + path1[i] / aveHigh;
                        std::cout << "\nH3 " << ntimepath1;
                        }
                }
        ntimepath1 = 60 * ntimepath1;

        ntimepath2 = 0;
        for (i = 0; i < length2; i++)
        {
                if (path2type[i] == 'C') {
                        ntimepath2 = ntimepath2 + path2[i] / aveCity;
                        std::cout << "\nC4 " << ntimepath2;
                }
                if (path2type[i] == 'I') {
                        ntimepath2 = ntimepath2 + path2[i] / aveInter;
                        std::cout << "\nI4 " << ntimepath2;
                }
                if (path2type[i] == 'H') {
                        ntimepath2 = ntimepath2 + path2[i] / aveHigh;
                        std::cout << "\nH4 " << ntimepath2;
                }
        }
        ntimepath2 = 60 * ntimepath2;
        std::cout << "\n" << ntimepath1 << " " << ntimepath2;
        if (ntimepath1 < ntimepath2)
                {
                std::cout << "\n the first route is faster.  It will take "
                << ntimepath1 << " minutes";
                }
                else { std::cout << "\n the second route is faster.  It will take "
                << ntimepath2 << " minutes";
                }
```

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

namespace FootballScoring.Advanced
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.Write("Enter score: ");
            int score = Convert.ToInt32(Console.ReadLine());
            int count = 0;
            for (int i = 0; i <= score; i += 2)
            {
                for (int j = i; j <= score; j += 3)
                {
                    for (int k = j; k <= score; k += 6)
                    {
                        for (int m = k; m <= score; m += 7)
                        {
                            if ((score - m) % 8 == 0)
                            {
                                count++;
                            }
                        }
                    }
                }
            }
            Console.WriteLine();
            Console.WriteLine("Number of ways: " + count);
            Console.ReadLine();
        }
    }
}
```