The Second Annual Community College Programming Contest: The Problems Used and the Problem Not Used

The Programming Contest Team
Everyone involved with the contest; including Bernice Houle, who took the lead in its organization and management, and Sylvester Tuohy, who took the lead in selecting the problems; insists the contest was a unified effort on the part of the Westchester Computer Science Department. And so it was.

The team is likewise unified in vigorously thanking its member Thomas Reivik, Seidenberg Network and Systems Analyst, for designing and implementing the technical infrastructure that made the contest possible as well as for his presence during the contest to ensure that all went well. All includes familiarizing the judges (CS faculty) and the contestants with his software, providing the judges with reviews as needed [frequently], and advising the contestants as questions arose on procedures of all kinds.
The 2nd Annual Community College Programming Contest
by Bernice Houle

The 2nd Annual Community College Programming Contest was held on Saturday, April 2, 2011 on the Pleasantville Campus. For four hours, 4 teams of students from the Hudson Valley Region worked diligently, many without a break, to work on 9 problems developed by the Westchester Computer Science faculty.

Each problem was worth 5 points. One point was deducted each time the problem was submitted with an error. If there was a tie score, then the time required to complete the problems was taken into consideration.

This year the Rockland Community College team won first place, snatching the reins from Orange County Community College who placed first in 2010. Congratulations go out to Robert Lee and Justin Okamoto, and their coach, Lynn Aaron. Robert and Justin successfully completed 6 questions with an overall score of 28 points. The grand prize was a $50 gift card awarded to each team member.

The 2nd Annual Community College Programming Contest was sponsored by the Westchester Computer Science Department with assistance from the Seidenberg Dean’s Office and support from a Thinkfinity grant from the Verizon Foundation.

---

2nd Annual Community College Programming Contest
(CCPC)
Sponsored by The Seidenberg School of
Computer Science & Information Systems at
Pace University (Pleasantville Campus)

Saturday, April 2, 2011

Solve Problems:
A, B, ... I

In the Shortest Time
Problem A

Encryption uses both substitutions and transformations and combinations of both. Here we will use only transformations. The encrypting transformation 1 2 3 4 5 6 7 → 5 3 1 4 6 7 2 maps the first letter to the fifth; second to the third etc. in 7 letter increments. As an example with no spaces being used PACEUNIVERSITY is encrypted to CIAEPUNRYESVIT.

The decrypting transformation is 1 2 3 4 5 6 7 → 3 7 2 4 1 5 6. And CIAEPUNRYESVIT is returned to the original text PACEUNIVERSITY.

Your problem is to use this decrypting transformation to find the original text. The cipher will be letters and numbers with no spaces and is <= 35 characters. Input is from the keyboard.

For example

Input:

Type the cipher. CIAEPUNRYESVIT

Output:

The plain text is. PACEUNIVERSITY

class ProblemA //prototype solution
{
    public static void main(String[] args)
    {
        String input = "CIAEPUNRYESVIT";

        while (input.length() >= 7)
        {
            System.out.print(input.charAt(4));
            System.out.print(input.charAt(2));
            System.out.print(input.charAt(0));
            System.out.print(input.charAt(3));
            System.out.print(input.charAt(5));
            System.out.print(input.charAt(6));
            System.out.print(input.charAt(1));

            input = input.substring(7);
        }
    }
}

Prob A: Test Data

MIOMCUNCEYOTLLPREROGMCMIAANGT2NEOST

Ans:
COMMUNITYCOLLEGEPROGRAMMINGCONTEST2

This is a simplified variation of Problem D from last year's contest on encryption with a transposition cipher. A block of plaintext ten characters long and the permutation used for encryption were entered from the terminal. The ciphertext was output.

This problem and a prototype solution may be found in Technical Report #274, dated May 2010.
Problem B

To triangulate is to locate a lost person with the use of three cell phone towers. The cell phone towers are the centers of a circle and the lost person is at some radius from each tower. One cell phone tower can indicate that the person is somewhere along the circumference of a circle at a given radius from the tower. Two cell phone towers can reduce the location to two points on the intersections of the circles. And the third cell phone tower decides which one of the two points is the correct location.

Our problem will give three centers and three radii of cell phone towers. All of the centers will be integer values in the Cartesian plane between (0, 0) and (20, 20) and the radii will also be an integer value. You are to find the location of the lost person which will have integer coordinates in this Cartesian plane. Prompt for three centers followed by their radii.

Input:

Input the center separated by a space. 1 2
Input the radius 13
Input the center separated by a space. 13 1
Input the radius 6
Input the center separated by a space. 17 10
Input the radius 5

Output:

Person is located at (13, 7)

```java
class ProblemB {  
    public static void main(String[] args) {  
        int r1 = 13, x1 = 1, y1 = 2; //tower1
        int r2 = 6, x2 = 13, y2 = 1; //tower2
        int r3 = 5, x3 = 17, y3 = 10; //tower3

        for (int x = 0; x <= 20; x++)  
        {  
            for (int y = 0; y <= 20; y++)  
            {  
                int pt1 = (x-x1)*(x-x1) + (y-y1)*(y-y1) - r1*r1;
                int pt2 = (x-x2)*(x-x2) + (y-y2)*(y-y2) - r2*r2;
                int pt3 = (x-x3)*(x-x3) + (y-y3)*(y-y3) - r3*r3;

                if (pt1 == 0 && pt2 == 0 && pt3 == 0)  
                {  
                    System.out.println("(" + x + ", " + y + ")");
                }
            }
        }
    }
}
```

Prob B: Test Data

<table>
<thead>
<tr>
<th>Center</th>
<th>Radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 15</td>
<td>13</td>
</tr>
<tr>
<td>7 7</td>
<td>5</td>
</tr>
<tr>
<td>2 10</td>
<td>1</td>
</tr>
</tbody>
</table>

Ans: Person located at (3,10)

<table>
<thead>
<tr>
<th>Center</th>
<th>Radius</th>
</tr>
</thead>
<tbody>
<tr>
<td>19 16</td>
<td>13</td>
</tr>
<tr>
<td>3 8</td>
<td>5</td>
</tr>
<tr>
<td>15 5</td>
<td>10</td>
</tr>
</tbody>
</table>

Ans: Person located at (7,11)

From analytic geometry: The formula for points on the circumference of a circle with its center at point (a, b) and having a radius r is:

\[(x - a)^2 + (y - b)^2 = r^2\]
Problem C

The problem is to count all of the occasions of a certain pattern. The input will be from the keyboard a string less than 80 characters followed by a pattern with no spaces to be found in the previous string. The match has to be case sensitive. The output will be the number of matches.

Examples:

Input:

Type in the String:

ASFGVTECCPCBHYCCPCCPCUPACEDFOHCCPCRSTM

Type in the pattern:

CCPC

Output:

The string "CCPC" has 5 matches.

Input:

Type in the String:

QWdfr45GthStrlnGvGSTRINGpoUYTFDstringMNOhyd

Type in the pattern:

StrlnG

The string "StrlnG" has 1 match.

Prob C: Test Data

Input String = CABCABbABbACABCABbACCBBABBBabba

Input the Pattern = ABbA

Ans: the String has 3 matches
class ProblemC //prototype solution
{
    public static void main(String[] args)
    {
        String text = "ASFGVTECCPCHYCCPCCPCUPACEDFOHCCPRSTM";
        String pattern = "CCPC";

        //String text = "QWdrf45GthStrINgVGSTRINGpoUYTFDStringMNOnyd";
        //String pattern = "StrINg";

        int lastIndex = text.length() - pattern.length();
        int index = 0;

        int instancesOfPattern = 0;

        while (index <= lastIndex)
        {
            String currentFragment = text.substring(index);
            System.out.println(currentFragment);

            if (currentFragment.startsWith(pattern))
            {
                instancesOfPattern++;
                System.out.println("match");
            }

            index++;
        }

        System.out.println(instancesOfPattern);
    }
}
Problem D

This problem is to compress the lines composing a black and white image. In the uncompressed image, 0 represents a black pixel, and 1 represents a white pixel. On a compressed line, single digit numbers alternate with the number of black and white pixels. The leading numeral represents the number of leading black pixels on the uncompressed line; it is zero if the leading pixel were white. See the first line in the sample input and output. When more than nine consecutive pixels are of the same color on an uncompressed line, the compressed line will show a '9' followed by a '0' and the color count will continue. See the second line in the sample input and output; it starts with eleven black pixels. Input from the keyboard is five lines of "twenty pixels" each. The output is the compressed image, one line of output for each line of input.

Example:

Input: (the uncompressed image)

Type in the image lines:

00000111110000011111
0000000000111111001
11111111111111111
010101010101010101
111111111100000000

Output: (the compressed image)

5555
902621
090902
1111111111111111
0901901

Prob D: Test Data

Input:

000000000011111111
1110001110011100110
000000111111100000
11001100110011001100
00000001010101111111

Ans: The Compressed File

901901
033323231
776
0222222222
71111117

```java
class ProblemD {
    static int index = 0;

    public static void main(String[] args) {
        String line1 = "00000000000111111001"; //902621
        String line2 = "11111111111111111"; //090902
        String line3 = "010101010101010101"; //11111111111111111
        String line4 = "111111111000000000"; //0901901
        String line5 = "00000111110000011111"; //5555

        process(line1); System.out.println();
        process(line2); System.out.println();
        process(line3); System.out.println();
        process(line4); System.out.println();
        process(line5); System.out.println();
    }
}
```
static void process(String line) //ProblemD prototype solution continued
{
    index = 0;

    while (index <= 19)
    {
        expectingBlack(line);
        if (index <= 19) expectingWhite(line);
    }
}

static void expectingBlack(String line)
{
    if (index == 0 && line.charAt(0) == '1')
    {
        System.out.print("0");
    }
    else if (line.charAt(index) == '0') //have black
    {
        //count the blacks
        int count = 0;
        while (index <= 19 && line.charAt(index) == '0') //lazy evaluation essential
        {
            count++;
            index++;
        }
        report(count);
    }
}

static void expectingWhite(String line)
{
    if (line.charAt(index) == '1') //have white
    {
        //count the whites
        int count = 0;
        while (index <= 19 && line.charAt(index) == '1') // lazy evaluation essential
        {
            count++;
            index++;
        }
        report(count);
    }
}

static void report(int count)
{
    if (count == 0) System.out.print("0");
    else
    {
        while (count >= 9)
        {
            System.out.print("9");
            count = count - 9;
            if (count > 0) System.out.print("0");
        }
        if (count > 0) System.out.print(count);
    }
}
Problem E

Given a bowl that contains some black beans and some white beans with extra black beans outside.

Do the following until a single bean is remaining:

Randomly select two beans from the bowl.

If they are of the same color throw them away and replace them with a black bean.

If they are of different colors throw the black away and replace the white bean to the bowl.

Your program should accept input from the keyboard for the number of black and the number of white beans (each less than 100). The output should be the color of the last bean. The program should have three trials.

For Example:

Type the number of black beans. 12
Type the number of white beans. 11
Output:
The last bean was white

Type the number of black beans. 12
Type the number of white beans. 20
Output:
The last bean was black

And this is repeated one more time.

Prob E: Test Data

Input: Number of black beans = 50
Number of white beans = 43
Ans = The last bean was white

Input: Number of black beans = 50
Number of white beans = 48
Ans = The last bean was black

Input: Number of black beans = 51
Number of white beans = 43
Ans = The last bean was white

Input an ODD number for white the answer is white else the answer is black.
class ProblemE  //prototype solution
{
    public static void main(String[] args)
    {
        int black = 12;
        int white = 20;

        int total = black + white;

        while(total > 1)
        {
            char r1 = random();
            char r2 = random();

            if (r1 == 'w' && r2 == 'w' && white >= 2)
            {
                white = white - 2;
                black = black + 1;
                total = black + white;
                System.out.println(r1+" "+r2+" blacks = "+black+" whites = "+white+//
                                " total = "+total);
            }
            else if (r1 == 'b' && r2 == 'b' && black >= 2)
            {
                black = black - 1;
                total = black + white;
                System.out.println(r1+" "+r2+" blacks = "+black+" whites = "+white+//
                                " total = "+total);
            }
            else if (white >= 1 && black >= 1)
            {
                black = black - 1;
                total = black + white;
                System.out.println(r1+" "+r2+" blacks = "+black+" whites = "+white+//
                                " total = "+total);
            }

        }

        if (white == 1 && black == 0)
        {
            System.out.println("last bean is white");
        }
        else if (white == 0 && black == 1)
        {
            System.out.println("last bean is black");
        }
    }

    static char random()  //returns 'b' for black, 'w' for white
    {
        if (Math.random() < .5)
            return 'b';
        else
            return 'w';
    }
}
Problem F

Find the maximum sum of contiguous numbers from a collection of 10. Input for your program should read from a file = “Problem F.dat” 10 two digit integers and output the largest contiguous sum.

For example:

Input:
23 45 -72 78 12 -21 89 -11 -45 33

Output:
158

From adding 78 to 89.

Prob F: Test Data

Input: -18 87 -45 -22 45 67 23 -98 12 34
Ans = 155 // adding 87 -45-22 45 67 23 (only the sum is necessary)

Input: -17 12 -48 13 -37 90 -22 10 -40 62
Ans = 100 // adding 90 -22 10 -40 62 (only the sum is necessary)
class ProblemF // prototype solution
{
    static int[] values = {23, 45, -72, 78, 12, -21, 89, -11, -45, 33};

    public static void main(String[] args)
    {
        int maxSum = Integer.MIN_VALUE;
        int vectorSubOfMax = -1;
        int startValue = 0;
        int endValue = 0;
        int startSub = -1;
        int endSub = -1; // span is the width of the interval
        for (int span = 9; span >= 1; span--) // span of 9 gets [0]..[9]
            // span of 8 gets [0]..[8] and [1]..[9]
            for (int i = 0; i < 10 - span; i++)
                {
                    int startSubscript = i;
                    int endSubscript = startSubscript + span;
                    int total = total(startSubscript, endSubscript);
                    if (total > maxSum)
                        {
                            maxSum = total;
                            startValue = values[startSubscript];
                            endValue = values[endSubscript];
                            startSub = startSubscript;
                            endSub = endSubscript;
                        }
                }

        System.out.println("[" + startSub + "]..[" + endSub + "], " + startValue + "." + endValue + ": " + maxSum);
    }

    static int total(int startSubscript, int endSubscript)
    {
        int sum = 0;

        for (int i = startSubscript; i <= endSubscript; i++)
            {
                sum = sum + values[i];
            }

        return sum;
    }
}
Problem G

The Roman Numeral Rules are the following:

\[ l = 1; \quad v = 5; \quad x = 10; \quad l = 50; \quad c = 100; \quad d = 500; \quad m = 1000 \]

a. A letter may be repeated up to 3 times \((xx = 20, \quad ccc = 300)\)
b. Smaller letters placed after a larger letter are added \((xvii = 17, \quad cclvi = 256)\)
c. A single smaller letter placed before a larger letter is subtracted \((ix = 9, \quad xl = 40)\)
d. The smaller letter (being subtracted) must be either the first letter or preceded by a letter at least ten times greater. \((dcd \neq 900 \quad \text{rather} \quad cm = 900)\)

Exceptions to Subtraction are:

Only subtract powers of 10 \((l, x, c \text{ are allowed but not } v \text{ and } l; \quad vl \neq 45 \quad \text{rather} \quad xlv = 45)\)

Do not subtract a number from one that is more than 10 times greater \((il \neq 49 \quad \text{rather} \quad xlxi = 49)\)

Your solution to the problem will ask for the conversion to Roman Numerals of three base 10 numbers.

Your program should prompt for the three numbers, one per line - the judges will test the program with three inputs. Maximum input from judges is 3000.

For Example:

Type first number. 83
Answer = LXXXIII

Type second number. 749
Answer = DCCXLIX

Type third number. 1024
Answer = MXXIV

<table>
<thead>
<tr>
<th>Prob G: Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type in the first number 494</td>
</tr>
<tr>
<td>Answer = CDXCIV</td>
</tr>
<tr>
<td>Type in the second number 2929</td>
</tr>
<tr>
<td>Answer = MMCMXXIX</td>
</tr>
<tr>
<td>Type in the third number 48</td>
</tr>
<tr>
<td>Answer = XLVIII</td>
</tr>
</tbody>
</table>

An Arabic-to-Roman and Roman-to-Arabic converter is at
[http://www.novaroma.org/via_romana/numbers.html](http://www.novaroma.org/via_romana/numbers.html)

A complete set of Arabic-to-Roman rules is at
class ProblemG // prototype solution page 1 of 6
{
    public static void main(String[] args)
    {
        // int number = 83; // LXXXIII
        // int number = 749; // DCCXLIX
        // int number = 1024; // MXXIV
        // int number = 2009; // MMIX
        // int number = 2858; // MMDCCLVIII
        // int number = 1858; // MDCCCLVIII
        // int number = 858; // DCCCLVIII
        // int number = 1947; // MCMXLVII
        // int number = 25; // XXV

        int[] test = {83, 749, 1024, 2009, 2858, 1858, 858, 1947, 25, 900};

        for (int i = 0; i < test.length; i++) test(test[i]);
    }

    static void test(int number)
    {
        System.out.print(number + " = ");

        String romanNumeral = "";

        // sifting off thousands ~and~ sifting off 100
        if (number >= 2900)
        {
            number = number - 2900;
            if (number >= 100)
            {
                romanNumeral = "MMM"; // 3000
                number = number - 100;
            }
            else
            {
                romanNumeral = "MMCM"; // 2900
            }
        }
        else if (number >= 1900)
        {
            number = number - 1900;
            if (number >= 100)
            {
                romanNumeral = "MM"; // 2000
                number = number - 100;
            }
            else
            {
                romanNumeral = "MCM"; // 2900
            }
        }
    }
}
else if (number >= 900)  //prototype solution
define number = number - 900;
if (number >= 100)
    {  
        romanNumeral = "M";  //1000  
        number = number - 100;
    }
else
    {  
        romanNumeral = "CM";  // 900
    }
}

//sifting off 800 to 100 ~and~ sifting off 10
if (number >= 800)
    {  
        number = number - 800;
        if (number >= 90)
            {  
                romanNumeral = romanNumeral + "DCCXC";  //890  
                number = number - 90;
            }
        else
            {  
                romanNumeral = romanNumeral + "DCCC";  //800
            }
    }
else if (number >= 700)
    {  
        number = number - 700;
        if (number >= 90)
            {  
                romanNumeral = romanNumeral + "DCCXC";  //790  
                number = number - 90;
            }
        else
            {  
                romanNumeral = romanNumeral + "DCC";  //700
            }
    }
else if (number >= 600)
    {  
        number = number - 600;
        if (number >= 90)
            {  
                romanNumeral = romanNumeral + "DCXC";  //690  
                number = number - 90;
            }
        else
            {  
                romanNumeral = romanNumeral + "DC";  //600
            }
    }
else if (number >= 500) //prototype solution
{
    number = number - 500;
    if (number >= 90)
    {
        romanNumeral = romanNumeral + "D"; //500
        number = number - 10;
    }
    else
    {
        romanNumeral = romanNumeral + "CD"; //400
    }
}
else if (number >= 400)
{
    number = number - 400;
    if (number >= 90)
    {
        romanNumeral = romanNumeral + "CDXC"; //490
        number = number - 90;
    }
    else
    {
        romanNumeral = romanNumeral + "CX"; //400
    }
}
else if (number >= 300)
{
    number = number - 300;
    if (number >= 90)
    {
        romanNumeral = romanNumeral + "CCCXC"; //390
        number = number - 90;
    }
    else
    {
        romanNumeral = romanNumeral + "CCC"; //300
    }
}
else if (number >= 200)
{
    number = number - 200;
    if (number >= 90)
    {
        romanNumeral = romanNumeral + "CCXC"; //290
        number = number - 90;
    }
    else
    {
        romanNumeral = romanNumeral + "CC"; //200
    }
}
else if (number >= 100) //prototype solution page 4 of 6
{
    number = number - 100;
    if (number >= 90)
    {
        romanNumeral = romanNumeral + "CX"; //90
        number = number - 90;
    }
    else
    {
        romanNumeral = romanNumeral + "C"; //100
    }
}
else if (number >= 90)
{
    romanNumeral = romanNumeral + "XC"; //90
    number = number - 90;
}

//sifting off 80 to 10 ~and~ 9
if (number >= 80)
{
    number = number - 80;
    if (number >= 9)
    {
        romanNumeral = romanNumeral + "LXXXIX"; //89
        number = number - 9;
    }
    else
    {
        romanNumeral = romanNumeral + "LXXX"; //80
    }
}
else if (number >= 70)
{
    number = number - 70;
    if (number >= 9)
    {
        romanNumeral = romanNumeral + "LXXIX"; //79
        number = number - 9;
    }
    else
    {
        romanNumeral = romanNumeral + "LXX"; //70
    }
}
else if (number >= 60) //prototype solution page 5 of 6
{
    number = number - 60;
    if (number >= 9)
    {
        romanNumeral = romanNumeral + "LXIX"; //69
        number = number - 9;
    }
    else
    {
        romanNumeral = romanNumeral + "LX"; //60
    }
}
else if (number >= 50)
{
    number = number - 50;
    if (number >= 9)
    {
        romanNumeral = romanNumeral + "LIX"; //59
        number = number - 9;
    }
    else
    {
        romanNumeral = romanNumeral + "L"; //50
    }
}
else if (number >= 40)
{
    number = number - 40;
    if (number >= 9)
    {
        romanNumeral = romanNumeral + "XLIX"; //49
        number = number - 9;
    }
    else
    {
        romanNumeral = romanNumeral + "XL"; //40
    }
}
else if (number >= 30)
{
    number = number - 30;
    if (number >= 9)
    {
        romanNumeral = romanNumeral + "XXXIX"; //39
        number = number - 9;
    }
    else
    {
        romanNumeral = romanNumeral + "XXX"; //30
    }
}
else if (number >= 20)  // prototype solution
{
    number = number - 20;
    if (number >= 9)
    {
        romanNumeral = romanNumeral + "XXIX";  // 29
        number = number - 9;
    }
    else
    {
        romanNumeral = romanNumeral + "XX";  // 20
    }
}
else if (number >= 10)
{
    number = number - 10;
    if (number >= 9)
    {
        romanNumeral = romanNumeral + "XIX";  // 19
        number = number - 9;
    }
    else
    {
        romanNumeral = romanNumeral + "X";  // 10
    }
}
else if (number >= 9)
{
    romanNumeral = romanNumeral + "IX";  // 9
    number = number - 9;
}

// sifting off 8 to 1
if (number == 8) romanNumeral = romanNumeral + "VIII";  // 8
else if (number == 7) romanNumeral = romanNumeral + "VII";  // 7
else if (number == 6) romanNumeral = romanNumeral + "VI";  // 6
else if (number == 5) romanNumeral = romanNumeral + "V";  // 5
else if (number == 4) romanNumeral = romanNumeral + "IV";  // 4
else if (number == 3) romanNumeral = romanNumeral + "III";  // 3
else if (number == 2) romanNumeral = romanNumeral + "II";  // 2
else if (number == 2) romanNumeral = romanNumeral + "I";  // 1

System.out.println(romanNumeral);
Shhhhh. Contestants hard at work....
Problem H

This problem is to find all the New York City ways to go. We are given a start location and a destination location as an ordered pair. A person has to walk always away from the source and towards the destination at every block. The streets in the problem will be in the range of 5th Ave. and 42nd to 9th Ave and 59th, without Broadway. There is no requirement on which way that we will travel (we do not necessarily have to move from a lower numbered street to a higher numbered street). Input will be from the keyboard - source and destination separated by a space. Output is only the number of paths and not what the paths are. Several cases will be checked.

As an example: There are 6 distinct ways to go from 0,0 to 2,2. Namely:

\[0,0 - 0,1 - 0,2 - 1,2 - 2,2; \quad 0,0 - 0,1 - 1,1 - 1,2 - 2,2; \quad 0,0 - 0,1 - 1,1 - 1,2 - 2,2\]

\[0,0 - 1,0 - 2,0 - 2,1 - 2,2; \quad 0,0 - 1,0 - 1,1 - 2,1 - 2,2; \quad 0,0 - 1,0 - 1,1 - 1,2 - 2,2\]

Input:

Type in the source separated by a space: 5 42
Type in the destination separated by a space: 5 50

Output:
The number of paths = 1

Input:

Type in the source separated by a space: 6 51
Type in the destination separated by a space: 8 49

Output:
The number of paths = 6

Input:

Type in the source separated by a space: 7 43
Type in the destination separated by a space: 5 57

Output:
The number of paths = 120

Prob H: Test Data

Input: 6 48 ➔ 5 50
Ans = The number of paths = 3

Input: 9 59 ➔ 7 43
Ans = The number of paths = 153

Input: 7 42 ➔ 7 57
Ans = The number of paths = 1

Input: 5 43 ➔ 7 46
Ans = The number of paths = 10
import java.util.*;  //prototype solution

public class ProblemH
{
    public static void main(String[] args)
    {

        Scanner keyboard = new Scanner(System.in);
        char repeat = 'y';
        while(repeat == 'y' || repeat == 'Y')
        {
            int sx = 0;  //source (x,y)
            int sy = 0;
            int dx = 0;  //destination (x,y)
            int dy = 0;
            int ct = 0;
            int temp = 0;
            System.out.println("Enter a source non-negative integer space integer : ");
            sx = keyboard.nextInt();
            sy = keyboard.nextInt();
            System.out.println("Enter a destination non-negative integer space integer : ");
            dx = keyboard.nextInt();
            dy = keyboard.nextInt();
            // System.out.println("sx ="+ sx + " sy = "+sy + "dx="+dx+"dy="+ dy);
            if (sx > dx) { temp = sx; sx = dx; dx = temp;}//swap
            if (sy > dy) { temp = sy; sy = dy; dy = temp;}//swap
            System.out.println("The number of paths are " + blocks(sx, sy, dx, dy, ct) );
            System.out.println();
            System.out.println("Do again? Enter 'y' for yes.");
            repeat = keyboard.next().charAt(0);
        }
    }

    private static int blocks(int u, int v, int dx, int dy, int c)
    {
        //System.out.println("u ="+ u + " v = "+v+ "dx="+dx+"dy="+ dy);
        //System.out.println("count = "+c);
        if(u == dx && v == dy) { c++; return c; }
        else if(u < dx) { c = blocks(u+1,v,dx,dy,c);}
        else if(v < dy) { c = blocks(u,v+1,dx,dy,c);}
        return c;
    }
}
Problem 1

Today (04/02/11) is signup for baseball. We have to have the age of the children (years, months and days) old to decide level upon which they will play. All children are born after 01/01/00 (month/day/year). Input from the keyboard will be the player’s name (first name only) and date of birth (month/day/year) in two digit format separated by slashes. Output will be the player’s name and the number of years, months, and days that have elapsed since birth.

Examples:

Input:
Enter the player: Michael
Enter the date of birth (mm/dd/yy): 01/05/04

Output:
Michael is 7 years 2 months and 28 days old today.

Input:
Enter the player: Catherine
Enter the date of birth (mm/dd/yy): 05/20/05

Output:
Catherine is 5 years 10 months and 13 days old today.

Input:
Enter the player: Genevieve
Enter the date of birth (mm/dd/yy): 04/02/00

Output:
Genevieve is 3 years 6 months and 2 days old today.

Problem 1 Test Data

Today's date is the date of the contest: 04/02/11

Input:
Enter the player: Kevin
Enter the date of birth (mm/dd/yy): 05/01/05

Output:
Kevin is 5 years 11 months and 1 day old today.

Input:
Enter the player: Elizabeth
Enter the date of birth (mm/dd/yy): 02/14/04

Output:
Elizabeth is 7 years 1 month and 19 days old today.

Input:
Enter the player: Brian
Enter the date of birth (mm/dd/yy): 04/02/00 //Happy Birthday!!!

Output:
Brian is 11 years 0 month and 0 days old today.
class ProblemI //prototype solution
{
    public static void main(String[] args)
    {
        String[] birthday = {"01/05/04", "05/20/05", "09/30/07", "05/01/05", "02/14/04", "04/02/00", "01/29/07", "02/29/07", "03/29/07", "04/02/07", "05/03/07", "11/10/07", "01/03/07", "02/04/07", "03/05/07", "01/01/07", "02/01/07", "04/01/07", "05/01/07", "06/01/07", "07/01/07", "12/01/07", "12/02/07", "12/03/07"};
        for (int i = 0; i < birthday.length; i++) age(birthday[i]);
    }

    static void age(String birthday) //convert bday String to ints
    {
        System.out.println(birthday + " = ");
        int month = Integer.parseInt(birthday.substring(0, 2));
        int day = Integer.parseInt(birthday.substring(3, 5));
        int year = Integer.parseInt(birthday.substring(6, 8));
        computeAge(month, day, year);
    }

    static void computeAge(int bdayMonth, int bdayDay, int bdayYear)
    {
        int todayMonth = 4, todayDay = 2, todayYear = 11; //date of contest: 04/02/11
        boolean birthdayAlreadyHappenedOrIsToday = birthdayAlreadyHappenedOrIsToday(bdayMonth, bdayDay, todayMonth, todayDay);

        int yearsOld = 0;
        if (birthdayAlreadyHappenedOrIsToday) yearsOld = todayYear - bdayYear;
        else yearsOld = todayYear - bdayYear - 1;

        int monthsOld = 0;
        int daysOld = 0;
        if (bdayMonth < todayMonth && bdayDay > todayDay) //e.g. bday = 02/29/07
        {
            monthsOld = todayMonth - bdayMonth - 1;
            daysOld = daysToEndOfMonth(todayMonth - 1, bdayDay, todayYear) + todayDay;
        }
        else if (bdayMonth >= todayMonth && bdayDay > todayDay) //e.g. today = 04/02/11
        {
            monthsOld = (12 - bdayMonth) + todayMonth - 1;
            daysOld = daysToEndOfMonth(bdayMonth, bdayDay, todayYear) + todayDay;
        }
        else if (bdayMonth < todayMonth && bdayDay <= todayDay) //e.g. today = 04/02/11
        {
            monthsOld = todayMonth - bdayMonth;
            daysOld = todayDay - bdayDay;
        }
    }
}
else if (bdayMonth >= todayMonth && bdayDay <= todayDay) // e.g. today = 04/02/11
    {
        if (bdayMonth == todayMonth && bdayDay == todayDay)
            {
                monthsOld = 0;
                daysOld = 0;
            }
        else
            {
                monthsOld = (12 - bdayMonth) + todayMonth;
                if (monthsOld == 12) monthsOld = 0;
                daysOld = todayDay - bdayDay;
            }
    }

System.out.println("Age: " + yearsOld + " year(s), " +
            monthsOld + " month(s), " + daysOld + " day(s)".reducer());

static int daysToEndOfMonth(int monthNumber, int day, int year)
    {
        int daysLeft = 0;
        switch(monthNumber)
            {
            case 1: case 3: case 5: case 7:  // 31-day month
            case 8: case 10: case 12:
                daysLeft = 31 - day;
                break;
                daysLeft = 30 - day;
                break;
            case 2:  // February
                if (year != 0 && year % 4 == 0) // this is a leap year
                    daysLeft = 29 - day;
                else
                    daysLeft = 28 - day; // not a leap year
                break;
            }
        return daysLeft;
    }
static boolean birthdayAlreadyHappenedOrIsToday
    (int bdayMonth, int bdayDay, int todayMonth, int todayDay )
    {
        boolean forReturn = true;
        if (bdayMonth < todayMonth)
        {
            forReturn = true;
        }
        else if (bdayMonth > todayMonth)
        {
            forReturn = false;
        }
        else if (bdayMonth == todayMonth && bdayDay <= todayDay)
        {
            forReturn = true;
        }
        else if (bdayMonth == todayMonth && bdayDay > todayDay)
        {
            forReturn = false;
        }
        return forReturn;
    }
}

/******************************************************************************/

C:\Contest-2011>java Problem1
01/05/04 = Age: 7 year(s), 2 month(s), 28 day(s)
05/20/05 = Age: 5 year(s), 10 month(s), 13 day(s)
09/30/07 = Age: 3 year(s), 6 month(s), 2 day(s)
05/01/05 = Age: 5 year(s), 11 month(s), 1 day(s)
02/14/04 = Age: 7 year(s), 1 month(s), 19 day(s)
04/02/00 = Age: 11 year(s), 0 month(s), 0 day(s)
01/29/07 = Age: 4 year(s), 2 month(s), 4 day(s)
02/29/07 = Age: 4 year(s), 1 month(s), 4 day(s)
03/29/07 = Age: 4 year(s), 0 month(s), 4 day(s)
04/02/07 = Age: 4 year(s), 0 month(s), 0 day(s)
05/03/07 = Age: 3 year(s), 10 month(s), 30 day(s)
11/10/07 = Age: 3 year(s), 4 month(s), 22 day(s)
01/03/07 = Age: 4 year(s), 2 month(s), 30 day(s)
02/04/07 = Age: 4 year(s), 1 month(s), 29 day(s)
03/05/07 = Age: 4 year(s), 0 month(s), 28 day(s)
01/01/07 = Age: 4 year(s), 3 month(s), 1 day(s)
02/01/07 = Age: 4 year(s), 2 month(s), 1 day(s)
04/01/07 = Age: 4 year(s), 0 month(s), 1 day(s)
05/01/07 = Age: 3 year(s), 11 month(s), 1 day(s)
06/01/07 = Age: 3 year(s), 10 month(s), 1 day(s)
07/01/07 = Age: 3 year(s), 9 month(s), 1 day(s)
12/01/07 = Age: 3 year(s), 4 month(s), 1 day(s)
12/02/07 = Age: 3 year(s), 4 month(s), 0 day(s)
12/03/07 = Age: 3 year(s), 3 month(s), 30 day(s)

C:\Contest-2011>

*******************************************************************************/
The 86th floor tourist observatory in the Empire State Building stands 1050 feet above the ground. How far can tourists see from there? Or (same question), looking out the window of an airplane from an altitude of 10,000 feet, how far is the horizon (where the earth's surface and the sky appear to meet)?

Write a program that accepts a height above ground level in feet and outputs in miles the vision range from that height. Ignore obstructions and terrain irregularities.

One mile equals 5,280 feet. The earth's diameter is 7926.28 miles.

For example:

Input:

Enter a height in feet:

Output:

From 1050 feet you can see 40.0 miles

class UnusedProblem    //prototype solution
{
    public static void main(String[] args)
    {
        int altitude = 1050;

        //Earth's diameter is 7926.28 miles
        double radius = (7926.28/2) * 5280;  //earth's radius in feet

        //Distance we can see from x feet of altitude can be found
        //using the Pythagorean Theorem:  a_squared + b_squared = c_squared
        double c = altitude + radius;       //the hypotenuse
        double c_squared = c * c;

        double a = radius;                  //one leg, the earth's radius
        double a_squared = a * a;

        //b is the other leg, the distance to the horizon -- how far you can see, in feet
        // (Horizon: the apparent line that separates earth from sky - Wikipedia)
        double b_squared = c_squared - a_squared;
        double b = Math.sqrt(b_squared);

        double distance = b/5280;           //distance in miles
        distance = Math.round(distance);
        System.out.println("From " + altitude + " feet you can see " + distance + " miles");
    }
}
The Seidenberg School of Computer Science and Information Systems
Pace University

Technical Report Series

EDITORIAL BOARD

Editor:
Allen Stix, Computer Science, Pace--Westchester

Associate Editors:
Constance A. Knapp, Interim Dean, The Seidenberg School of Computer Science and Information Systems – Pace University
Susan M. Merritt, Computer Science, Pace--Westchester

Members:
Howard S. Blum, Computer Science, Pace--New York
Mary F. Courtney, Computer Science, Pace--Westchester
Nicholas I. De Lillo, Mathematics and Computer Science, Manhattan College
Daniel Parkas, Information Systems, Pace--Westchester
Fred Grossman, Information Systems; Doctor of Professional Studies, Pace--New York and White Plains
Fran Goertzel Gustavson, Information Systems, Pace--Westchester
Joseph F. Malerba, Computer Science, Pace--Westchester
John S. Mallozzi, Computer Information Sciences, Iona College
John C. Molluzzo, Information Systems, Pace--New York
Pauline Mosley, Technology Systems, Pace--New York
Narayan S. Murthy, Computer Science, Pace--New York
Catherine Ricardo, Computer Information Sciences, Iona College
Judith E. Sullivan, CSIS Alumna, MS in CS from Pace--Westchester
Sylvester Tuohy, Computer Science, Pace--Westchester

The Seidenberg School of Computer Science and information Systems, through the Technical Report Series, provides members of the community an opportunity to disseminate the results of their research by publishing monographs, working papers, and tutorials. Technical Reports is a place where scholarly striving is respected.

All preprints and recent reprints are requested and accepted. New manuscripts are read by two members of the editorial board; the editor decides upon publication. Statements of policy and mission may be found in issues #29 (April 1990) and #34 (September 1990).

Please direct submissions as well as requests for single copies to:

Allen Stix
The Seidenberg School of Computer Science and Information Systems
Goldstein Academic Center
Pace University
861 Bedford Road
Pleasantville, NY 10570-2799