First Annual Hood College
High School
Programming Contest 2005

- **Problem A:** Ultimate Collapse
- **Problem B:** Moldy Chocolate
- **Problem C:** Hot Potato
- **Problem D:** All You Need is Love!
- **Problem E:** Concurrency
- **Problem F:** Special Words
- **Problem G:** Wally's Traversals
- **Problem H:** Clockwise or Counter-Clockwise?
- **Errata**

Special Note: C++ and Java programs should provide an appropriate prompt for input. Please be certain that the prompt is not ambiguous or vague. For Visual Basic.Net programs, input may be read in from a form; please clearly label what you expect from the user.

Problem A
Ultimate Collapse

Introduction: Given an integer n, we define its collapse as the sum of its digits. We can then take the sum of the digits of the collapse of n and obtain the super-collapse of n. Repeating the procedure yet again produces the super-duper collapse of n. We can repeat the procedure over and over again until we get a single digit. This single digit is known as the ultimate collapse of n.

For example, consider n = 15827. The sum of the digits of n is 23; this is the collapse of n. The sum of the digits of 23 is 5. Since 5 is a single-digit number, we conclude that the ultimate collapse is 5.

Problem Statement: Write a program that allows the user to enter a positive integer n. The program then outputs the ultimate collapse of n. Note that n may range from 1 to one billion.

Sample Input	Sample Output
n = 15827	Ultimate collapse is 5.
n = 2879014	Ultimate collapse is 4.

Submit to:

- JudgeA@pluto.hood.edu, if you write a program in Java or C++.
- JudgeJ@pluto.hood.edu, if you write a program in VB.Net.

Problem B
Moldy Chocolate

Introduction: The Swiss are dead serious about their chocolate. In fact, they're so serious that they are the international champions at the Olympic sport known as "Moldy Chocolate"

Here's how the sport is played. Two players sit at a table, with M pieces of chocolate placed in front of them. Exactly one of these pieces is moldy; the other pieces are made of the finest Swiss chocolate. The two players alternate taking turns. During each turn, a player eats anywhere from between 1 and N chocolate pieces, avoiding the moldy piece whenever possible. The player who gets stuck eating the moldy chocolate is declared the loser.

For example, suppose M=10 and N=2. Then there are 10 pieces of chocolate: 1 is moldy, while the other 9 are good. Also, each player eats between 1 and 2 pieces of chocolate per turn. Suppose, on his first turn, Player A decides to eat 2 pieces of chocolate, leaving 8 pieces. Player B likewise decides to eat 2 pieces, leaving 6 pieces. Player A then eats 1 piece, and then player B eats 2 pieces. At this point, there are 3 pieces left. Suppose Player A eats 1 piece, and then Player B eats 1 piece. This leaves the final piece--the moldy chocolate -- for Player A to eat. Player A loses, and so Player B is declared the winner.

Problem Statement: Write a program that allows the user to enter M and N. The program then outputs the winner of Moldy Chocolate, assuming each player adopts an optimal strategy (i.e. plays to win). You should assume that Player A goes first.

Finally, M and N can range anywhere from 1 to one billion.

Sample Input	Sample Output
M = 13, N = 2	Player B wins.
M = 24, N = 5	Player A wins.

Submit to:

- JudgeB@pluto.hood.edu, if you write a program in Java or C++.
- JudgeJ@pluto.hood.edu, if you write a program in VB.Net.

Problem C
Hot Potato

Introduction: A popular game in Idaho is Hot Potato.

Here's how the game is played. There are M players who are assigned the numbers 1 through M. They arrange themselves in a circle, facing inward, and ordered in a clockwise fashion. There are M-1 rounds throughout the game. At the end of each round, one player is eliminated. In the end, there is only one player left standing who is declared the winner.

The rounds proceed as follows: At the beginning of the game (the first round), Player 1 picks up a hot potato and screams "Ow! Ow! Too hot!", then quickly tosses the hot potato to his left (namely, to Player 2). Player 2 then repeats the exact same phrase, and quickly tosses the potato to Player 3. This continues on like this until the Nth person, who, unfortunately, is forced to eat the hot potato, scorch the roof of his mouth, and is eliminated from the game. The next round then starts with the player who is to the left of the player that was just eliminated and the process is repeated.

For example, suppose M = 5 and N = 3. So there are 5 players and, during each round, the 3rd player is forced to eat the hot potato and is thus eliminated. The players are eliminated in the following order: 3,1, 5, and 2. So Player 4 is the winner.

Problem Statement: Write a program that allows the user to enter M and N. The program should then output the winning player.

Sample Input	**Sample Output**
M = 5, N = 3	Player 4 wins!

Submit to:

- JudgeC@pluto.hood.edu, if you write a program in Java or C++.
- JudgeJ@pluto.hood.edu, if you write a program in VB.Net.

Problem D
All You Need is Love

"All you need is love. All you need is love.
All you need is love, love... love is all you need."
The Beatles

Introduction: There was invented a new powerful gadget by the International Beautiful Machines (IBM) corporation called the love machine! Given a string made of binary digits, the love machine answers if it's made only of love, that is, if all you need is love to build that string. The definition of love for the love machine is another string of binary digits, given by a human operator. Let's say we have a string L which represents "love" and we give a string S for the love machine. We say that all you need is love to build S, if we can repeatedly subtract L from S until we reach L. The subtraction defined here is the same arithmetic subtraction in base 2. By this definition it's easy to see that if L > S (in binary), then S is not made of love. If S = L then S is obvious made of love.

Let's see an example. Suppose S = "11011" and L = "11". If we repeatedly subtract L from S, we get: 11011, 11000, 10101, 10010, 1111, 1100, 1001, 110, 11. So, given this L, all you need is love to build S. Because of some limitations of the love machine, there can be no string with leading zeroes. That is, "0010101", "01110101", "011111" etc. are invalid strings. Strings which have only one digit are also invalid (it's another limitation).

Problem Statement: Your task in this problem is: have the user input two valid binary strings, S1 and S2. Your program should determine if it's possible to have a valid string L such that both S1 and S2 can be made only of L (i.e. given two valid strings S1 and S2, find if there exists at least one valid string L such as both S1 and S2 are made only of L). For instance, for S1 = 11011 and S2 = 11000, we can have L = 11 such that S1 and S2 are both made only of L (as we can see in the example above).

Note that the binary strings S1 and S2 can each be as long as 30 bits.

Sample Input	**Sample Output**
S1 = 11011, S2 = 11000	All you need is love!
S1= 11011, S2 = 11001	Love is not all you need!
S1 = 111111, S2 = 100	Love is not all you need!
S1 = 1010, S2 = 100	All you need is love!

Submit to:

- JudgeD@pluto.hood.edu, if you write a program in Java or C++.
- JudgeJ@pluto.hood.edu, if you write a program in VB.Net.

Problem E
Concurrency

Introduction: One of the goals of distributed computing is to take a set of tasks and schedule them amongst different processors. As an analogy, one can imagine an entire list of chores (e.g. cooking, cleaning, walking the dog, etc.) that could be distributed amongst different people (e.g.Dad goes grocery shopping, Mom does the cooking, etc.) However, things can get very tricky since, typically, one cannot distribute the tasks in an arbitrary fashion. Certain tasks must usually precede other tasks (e.g. Dad has to do the grocery shopping before Mom can cook dinner.). In fact, given a pair of tasks T1 and T2 we say that T1 and T2 are *non-concurrent* if one of them must precede the other. Otherwise, we say that they are *concurrent.*

Things can get even trickier. For instance, suppose task T1 must precede task T2, and task T2 must precede task T3. Then, by necessity, task T1 and T3 are non-concurrent; in fact, we can conclude that T1 must precede T3. This property is known as *transitivity*..

Problem Statement: Write a program that allows the user to enter a sequence of pairs of tasks. Within each pair, the first is a prerequisite task for the second. For simplicity, we assume that there are exactly 26 tasks that we are concerned with, and they are labeled by lower-case letters. Thus, we can express such a list of course pairings by a string.

For example, if task c is a prerequisite to task m, task m is a prerequisite to task k, and task z is a prerequisite to task q, then such a scenario would be represented by the string cmmkzq.

Your program should then output the number of pairs of tasks that are concurrent.

Sample Input	**Sample Output**
S = ”cmmkzq”	

Submit to:

- JudgeE@pluto.hood.edu, if you write a program in Java or C++.
- JudgeJ@pluto.hood.edu, if you write a program in VB.Net.

Problem F
Special Words

Introduction: Sally thinks that certain words are special. For instance, "accent" is special because as one reads it left-to-right, the letters are all in alphabetical order. (We consider repeated letters, such as the "cc", as still in alphabetical order.) However, "acid" is not special since, the "i" and the "d" are not in alphabetical order.

Sally loves special words so much that she always wants to see special-ness in any word that she can. As an example, when faced with the word "alphabetical", she attempts to eliminate some letters in order to produce the longest possible special word, even if the remaining letters make a nonsense word. In this case, the longest possible special word she can extract from "alphabetical" consists of five letter (e.g. "aabcl" or "aabel" or "abeil".).

Problem Statement: Write a program that allows the user to enter a word W. The program should then output the length of the longest special word that can be extracted from W. Note that the maximum length of the word W is 30 letters long.

Sample Input	**Sample Output**
W = "alphabetical"	The longest special word has length 5.

Submit to:

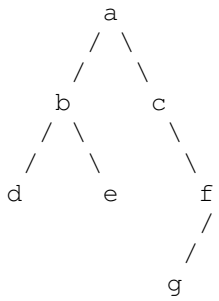
- JudgeF@pluto.hood.edu, if you write a program in Java or C++.
- JudgeJ@pluto.hood.edu, if you write a program in VB.Net.

Problem G
Wally's Traversals

Introduction: Wally the Worm is a fitness nut. Every day, he takes a walk around a rooted binary tree. He begins by positioning himself to the left of the root node. He then proceeds to walk along the outside edge of the tree, tracing out its shape along the way:

Of course, there are three ways of describing his route: preorder traversal, postorder traversal, and inorder traversal. The preorder (respectively, postorder and inorder) traversal is found by recording the name of the node as Wally passes on its left (respectively, right and below). Typically, Wally keeps a journal of his walk by recording all three traversals. But, on occasion, he loses track of one of the traversals and often has to reconstruct it from the other two.

For instance, one day, Wally walked around the following rooted binary tree:



He recorded his preorder traversal as abdecfg and the inorder traversal as dbeacgf. Unfortunately, Wally neglected to record the postoreder traversal, but, luckily, he was able to reconstruct it: debgfca.

Problem Statement: Given a rooted, binary tree T, write a program that allows the user to enter the preorder traversal of T and the inorder traversal of T. The program should then output the postorder traversal of T.

You may assume that each node is represented by a character. So a traversal (whether preorder, postorder, or inorder), is expressed as a string. The string can be as long as 26 characters (the number of lower-case letters.)

Sample Input	**Sample Output**
Preorder = "`abdecfg`", Inorder = "`dbeacgf`"	Postorder = "debgfca"

Submit to:

- JudgeG@pluto.hood.edu, if you write a program in Java or C++.
- JudgeJ@pluto.hood.edu, if you write a program in VB.Net.

Problem H
Clockwise or
Counter-Clockwise?

Introduction: Given a sequence of three non-colinear points, we can form a triangle and determine if these three points list the vertices of the triangle in clockwise or counter-clockwise order. For instance, if the three points are (0,0), (1,0), and (0,1) (in that order), then we can determine that these three points are given in counter-clockwise order. However, the sequence (0,0), (0,1), and (1,0) is in clockwise order.

Problem Statement: Write a program that allows the user to enter three points P1, P2, and P3. The program should then output the message "clockwise" or "counterclockwise" as appropriate. You may assume that the three points are not collinear.

Sample Input	Sample Output
P1 = (0,0), P2 = (1,0), P3 = (0,1)	Counter clockwise
P1 = (0,0), P2 = (0,1), P3 = (1,0)	Clockwise

Note: it's OK if you ask for each coordinate separately. Your program should make it obvious as to which point is P1, P2, and P3 and which coordinate (x or y) is being asked for.

Submit to:

- JudgeH@pluto.hood.edu, if you write a program in Java or C++.
- JudgeJ@pluto.hood.edu, if you write a program in VB.Net.

Errata

The following corrections were made during the contest:

- For Problem E, the sample output was missing. It should read "There are 321 concurrent events."
- For Problem F, the sample output should read "The longest special word has length 6".
- For Problem H, the last sentence in the introduction should read "However, the sequence (0,0), (1,0), and (0,1) is in clockwise order." [Note: no one seemed to notice this one. Probably, because the correct answer was listed as one of the sample outputs, it didn't lead to any confusion.]