# Fifth Annual Quinnipiac University Programming Competition for High School Students

Quinnipiac University

March 28, 2009

## Instructions

1. Submit a program by attaching all necessary files to an e-mail to `cscjudge`. Include the number of the problem you are solving in the subject line of your e-mail.

   Questions may also be addressed to `cscjudge`; put the word "Question" in the subject line, along with the number of the question you are asking about if your question pertains to a particular problem.

2. You may assume the input is formatted as shown in the problem, and your program must work on such input. You may assume there are no tabs in the input.

3. Your feedback on a submitted problem will be e-mailed in a reply and will be limited to one of the following:

   (a) COMPILE ERROR: The program did not compile

   (b) PROGRAM NOT RUNNABLE: The program compiled, but did not run as a stand-alone program.

   (c) CRASH: The program crashed or threw an exception when it was run

   (d) TIMED OUT: The program did not finish in a reasonable amount of time

   (e) INCORRECT OUTPUT: Either the answer is incorrect, or it is not presented in the form specified in the problem.

   (f) ACCEPTED

   If your program crashes, produces incorrect output, or times out, no information will be given as to the input that caused the problem.

# 1 Power sums

Write a program which takes two integers $n$ and $k$ on separate lines as input ($n$ first), and prints the sum of the $k$th powers of all the integers between $1^k$ and $n^k$, inclusive. You may assume that $1 \leq n \leq 10$ and $0 \leq k \leq 9$. For example, the input

```
10
9
```

asks for the sum of all ninth powers between $1^9$ and $10^9$. The output in this case should be `1574304985`, since

$$1^9 + 2^9 + 3^9 + 4^9 + 5^9 + 6^9 + 7^9 + 8^9 + 9^9 + 10^9 = 1,574,304,985.$$

Similarly, the input

```
6
2
```

should produce the output $91$, since $1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 = 91$. The input

```
5
0
```

should produce the output $5$, since any number raised to the zeroth power is $1$.

## 2 It's About Time

Write a program which takes as input the current time and a number of minutes to add to or subtract from the current time, and computes the corresponding time in the future or the past. The input consists of three integers on one line. The first integer is between $1$ and $12$, inclusive, and represents the current hour on a twelve-hour clock. The second is between $0$ and $59$, inclusive, and represents the current minute. The last number gives a number of minutes to add or subtract: a positive number represents minutes in the future, and a negative number represents minutes in the past. You may assume this number is between $-525,600$ and $525,600$ inclusive (the number of minutes in a year). The output should be the new time, including the hour between $1$ and $12$ followed by a colon and the minute. If the minute is less than ten, make sure to include an extra zero in the output, i. e. "`12:05`" instead of "`12:5`."
For example, if we want to know what time a contest that starts at 9:30 and lasts for three hours ($180$ minutes) will end, we could use the following input:

```
9 30 180
```

the output should be `12:30`. If the contest lasts four hours ($240$ minutes), we would use the input

```
9 30 240
```

and the answer would be `1:30`. Finally, if we the current time is 1:20 and we want to know what time it was $135$ minutes ago, we could use the input

```
1 20 -135
```

to get the output `11:05`.

# 3 Magic Squares

A magic square is a square grid of numbers with the property that the sum of the numbers in each row, each column, and both diagonals is the same. For example, the grid

| | | |
|---|---|---|
| 8 | 1 | 6 |
| 3 | 5 | 7 |
| 4 | 9 | 2 |

is a magic square since the sum of the numbers in each row ($8+1+6$; $3+5+7$; and $4+9+2$), column ($8+3+4$; $1+5+9$; and $6+7+2$), and both diagonals ($8+5+2$ and $6+5+4$) are all $15$. Write a program which reads in a square grid of integers and prints either the word "yes" if the grid represents a magic square and "no" otherwise. The input starts with the number of rows and columns on a line by itself, which we call $n$. It then has $n$ rows, each with $n$ integers separated by one or more spaces representing the entries in the grid. For example, the input for the magic square given above could be

```
3
8 1 6
3 5 7
4 9 2
```

You may assume $n \leq 9$, and every entry is between $-100$ and $100$. The integers may be positive, negative, or zero, and they may or may not be distinct.

# 4 Minesweeper

The game Minesweeper is played on a rectangular board on which some of the squares contain mines. A player attempts to uncover all the unmined squares by repeatedly choosing squares to uncover. When a player steps on a safe square, the game shows the number of mines on neighboring squares (including squares that are diagonally adjacent to the chosen square). Of course, if a player chooses a square with a mine the game is over and the player loses.
Write a program which computes for each safe square on a minesweeper board how many of its neighbors have mines. The input consists of two integers giving the number of rows and columns in the grid, followed by the board that gives the configuration of the mines. On this board, a period represents a safe square and an asterisk represents a mine. Your output should reprint this board, but with each period replaced with the number of neighboring mines.
For example, if the input is

```
8 20
.*.*...*.......**.**
.*.*.*....*.*.*.*.**
...*.*.*..*...**..*.
**...*.*..*...*...*.
*.....*.*..**....*..
***.*.*.*..*...*....
*...*......*...***..
.****..*..**.*******.*
```

the output should be

```
2*4*312*1112122**4**
2*5*5*4222*3*3*6*5**
334*5*5*23*414**24*4
**213*5*33*423*323*2
*63224*5*33**2222*21
***3*4*4*23*412*5320
*655*423234*434***21
2***22*11**3***3322*
```

where, for example, the entry in the top left corner is a two because there are mines both on the square immediatly to the right, and diagonally below and to the right of the corner square.
You may assume both the number of rows and columns are between $2$ and $60$, inclusive.

# 5 An Error-correcting Code

When we send a message over a network, we generally have to consider the possibility that there is some "noise" that can disrupt the message. That is, a zero that is sent may be changed to a one or *vice versa*. A typical solution is to send a message that includes only certain sequences of bits, called *codewords*. Then if a different message is received the error can be detected and in many cases, even corrected. The receiver simply assumes the intended message is the codeword with the fewest differences from the received sequence of bits.
An *error-correcting code* is a collection of codewords with the property that a single error can always be corrected. One example of an error-correcting code consists of the following 16 seven-bit codewords, called a *Hamming code* after its inventor Richard Hamming.

0000000 0000111 0011001 0011110 0101010 0101101 0110011 0110100
1001011 1001100 1010010 1010101 1100001 1100110 1111000 1111111

In fact, this code is called a *perfect code* because every sequence of seven bits that is not one of these codewords can be corrected to a codeword by changing exactly one bit, and the corrected word is unique. We call any sequence of seven bits a *word* whether it is a codeword or not.
These codewords are the only seven-bit words with the following properties:

1. There are an even number of ones among the first, third, fifth, and seventh bits.
2. There are an even number of ones among the first, second, fifth, and sixth bits.
3. There are an even number of ones among the first four bits.

For example, 0011001 has two ones among the first, third, fifth, and seventh bits, it has zero ones among the first, second, fifth, and sixth bits, and two ones among the first four bits.
The error-correction works because each of the seven bits affects a different set of the above properties. For example, if we receive the word 0001001, Properties 1 and 3 fail and property 2 holds. Therefore, if we assume only one bit was incorrect, that bit must be counted in the first and third groups, but not the second. The only such bit is the third bit, so changing that, we assume the word that was sent was 0011001.
Write a program which takes a list of words, each on a line by itself, followed by a line of seven `x`'s. Your program should print the word "`correct`" for each codeword, and print the corrected codeword for a word that is not in the code.
For example, if the input is

```
0011001
0001001
0000001
xxxxxxx
```

the output should be

```
correct
0011001
0000000
```

You may print the result for one word before reading in the next one.

# 6 Spelling numbers

Write a program which takes an integer (positive, negative, or zero) whose absolute value is less than one billion as input. The program then spells the number in English according to the following rules:

1. All answers should be in lowercase.
2. All words in your answer should be taken from the list at the bottom of the page and spelled accordingly. The word "and" never appears in the answer.
3. The number of millions, thousands and others should all appear on separate lines. All lines after the first should be indented two spaces (*exactly*).
4. Negative numbers should begin with the word "negative" on the first line.
5. If the number is between $21$ and $99$ and the units digit is not zero, there should be a hyphen between the words for the two digits, e. g. "`twenty-one`" and similarly there should be hyphens for the same numbers of thousands and millions, e. g. "`twenty-one million three hundred forty-five thousand`." There should be a single space and no hyphens between all other words.
6. If your answer is misspelled, mispunctuated, or includes uppercase letters, or if any spaces are added or missing, your answer is wrong.

For example, if the input is

```
-123000018
```

the output should be

```
negative one hundred twenty-three million
  eighteen
```

The following list consists of all words used in answers, properly spelled. Check your spelling carefully.

```
zero      one      two        three     four
five      six      seven      eight     nine
ten       eleven   twelve     thirteen  fourteen
fifteen   sixteen  seventeen  eighteen  nineteen
twenty    thirty   forty      fifty     sixty
seventy   eighty   ninety
negative  hundred  thousand   million
```

# 7 Solving two Equations in two Unknowns

A common problem is to find values $x$ and $y$ that satisfy two different equations simultaneously. For example, if the equations are

$$3x + 2y = 8$$
$$2x - y = 6$$

the solution is $x = \frac{20}{7}$ and $y = -\frac{2}{7}$, since $3\left(\frac{20}{7}\right) + 2\left(-\frac{2}{7}\right) = 8$ and $2\left(\frac{20}{7}\right) - \left(-\frac{2}{7}\right) = 6$.
The two equations are called a *system of equations*, and finding a solution that satisfies both of them is called *solving the system of equations*. The variables whose values we are trying to solve for are called *unknowns*.
In some cases, it may be impossible to solve both equations, and in others, there may be infinitely many solutions. For example, it is impossible to solve the following system of equations:

$$3x + 2y = 8$$
$$6x + 4y = 10$$

However, there are infinitely many solutions to this system:

$$4x - 2y = 12$$
$$2x - y = 6$$

since given any $x$, we have a solution when $y = 2x - 6$.
The general form for such a system is

$$ax + by = m$$
$$cx + dy = n$$

Write a program which reads in the values of $a$, $b$, and $m$ on one line and $c$, $d$, and $n$ on the next (all integers), with one or more spaces between values on each line. The program should print out the unique solution if there is one, or the words "`no solution`" or "`many solutions`" if either of those statements is appropriate. If there is a unique solution, your program should print out a line of the form

```
x=20/7; y=-2/7
```

i. e. "`x=`" followed by the value of $x$, then a semicolon and one space, then "`y=`" and the value of $y$. The fraction must be in lowest terms, and the denominator must be positive.
Here are some examples of input and output.

| **Input** | `3  2 8`<br>`2 -1 6` | `3 2  8`<br>`6 4 10` | `4 -2 12`<br>`2 -1 6` | `3 -2  4`<br>`9 4 2` |
|---|---|---|---|---|
| **Output** | `x=20/7; y=-2/7` | `no solution` | `many solutions` | `x=2/3; y=-1` |

You may assume that all the integers in the problem are between $-1000$ and $1000$, inclusive, but the numerators and denominators of the solutions may be larger. Note there is no denominator for $y$ in the last example, since the value is an integer.

# 8 Traveling circus

The manager of a traveling circus has agreed to visit a number of locations, and he plans to have the circus return home after the shows. He has not yet decided in what order to visit each location. He wants to order of the visits in such a way that he minimizes the distance the circus has to travel, including the last trip home. Write a program that plans the route. The input to the program begins with an integer $n$ on a line by itself representing the number of locations (including home), followed by the name of each place, each on a line by itself, followed by an $n \times n$ grid of nonnegative numbers representing the distances from each location to every other location.

For example, the following input gives all the distances between eight institutions: Quinnipiac University and seven high schools.

```
8
Quinnipiac University
Darien High School
North Haven High School
Greater Hartford Academy
Tolland High School
Staples Academy
Norwich Free Academy
Barlow High School
 0 44  5 31 48 36 62 37
44  0 43 74 91 13 94 24
 5 43  0 33 51 35 64 36
31 74 33  0 23 65 41 58
48 91 51 23  0 82 32 81
36 13 35 65 82  0 84 11
62 94 64 41 32 84  0 88
37 24 36 58 81 11 88  0
```

Since Darien High School is the second school in the list and Staples Academy is the sixth, the distance between these schools is given by the sixth entry in the second row (or the second entry of the sixth row); in this case, it is $13$ miles.

Your program should give a route that minimizes the total distance the circus has to travel, starting and ending in the first location on the list. Each location should be on a line by itself, followed by a space, and then by the total mileage in the route thus far in parentheses. If two routes have the same total distance, choose the route in which the first city where the routes differ comes earlier on the list. The output for the above input is given on the following page.

The following is the sample input from the previous page.

```
8
Quinnipiac University
Darien High School
North Haven High School
Greater Hartford Academy
Tolland High School
Staples Academy
Norwich Free Academy
Barlow High School
 0 44  5 31 48 36 62 37
44  0 43 74 91 13 94 24
 5 43  0 33 51 35 64 36
31 74 33  0 23 65 41 58
48 91 51 23  0 82 32 81
36 13 35 65 82  0 84 11
62 94 64 41 32 84  0 88
37 24 36 58 81 11 88  0
```

With this input, your program should produce the following output.

```
Quinnipiac University (0)
North Haven High School (5)
Barlow High School (41)
Staples Academy (52)
Darien High School (65)
Norwich Free Academy (159)
Tolland High School (191)
Greater Hartford Academy (214)
Quinnipiac University (245)
```

Note that following the same route in reverse, from Quinnipiac to Greater Hartford Academy to Tolland High School and so on would produce the same total distance, but the program picks the route that goes to North Haven High School first since North Haven High School precedes Greater Hartford Academy in the list of locations.

You may assume that there are at most $9$ locations, including the home location, and the distance between any pair of locations is at most $3000$ miles. You may also assume that the distance from any location to itself is $0$ miles, and the distance from the $i$th location to the $j$th location is the same as the distance from the $j$th location to the $i$th location.

# 9 Medical Residencies

When a prospective doctor finishes medical school, s/he first serves as a resident in a hospital. The hospitals are given information about every graduating student and the students are given information about the available hospitals. Each hospital ranks the students, and each student ranks the hospitals. These rankings are then submitted to a central office that matches the students with the hospitals.

To do this matching, the central office simulates the following process: each hospital offers its residency to its top ranking student. If a student has more than one offer, s/he rejects all offers except the one s/he ranks the highest. The remaining offer is neither accepted nor rejected yet. Then, each hospital whose latest offer has been rejected offers the residency to its highest ranking remaining student. This process continues until every hospital has an offer outstanding, at which point every student accepts the current offer.

Write a program that does the matching process. In our version, we assume there are $n$ students and $n$ residencies available, (i. e. the same number of each), with $n \leq 10$. Each student and hospital are numbered from $1$ to $10$ to preserve their anonymity. The program takes as input the number $n$ on a line by itself, followed by the preferences for each student, and then the preferences of each hospital. For example, consider the following input, with three students and three hospitals:

```
3
2 1 3
3 2 1
1 3 2
3 2 1
1 3 2
2 1 3
```

This means the first student prefers, in order, hospital 2, then hospital 1, then hospital 3, and the first hospital prefers, in order, student 3, then student 2, then student 1, (its preferences appear after those of the three students).

The output should consist of $n$ lines which each have two numbers separated by a hyphen. The $k$th line of output has the number $k$ and a hyphen followed by the number of the hospital matched with the $k$th student. For example, given the input above, hospital 1 asks student 3, hospital 2 asks student 1, and hospital 3 asks student 2. The students are all immediately spoken for, so they accept the offers, and the output should be

```
1-2
2-3
3-1
```

For a more complicated example, see the next page.

If the input is as follows (where we added a line to separate the students rankings from those of the hospitals)

```
6
5 3 4 1 2 6
2 1 6 3 5 4
4 1 3 6 2 5
1 5 2 3 6 4
5 6 2 3 4 1
4 1 5 6 3 2
```

---

```
5 1 6 4 3 2
1 6 3 4 5 2
6 4 5 1 3 2
4 5 1 6 3 2
3 6 4 1 5 2
1 4 6 3 5 2
```

the corresponding output is

```
1-5
2-2
3-3
4-1
5-6
6-4
```