University of Arkansas
Department of Computer Science and Computer Engineering
2017 High School Programming Contest
Problems

Wing Ning Li Hung Nguyen Adam Higgins Paul Martin Tyler Moore
Matthew Luther

Revision Date: March 9, 2017

Please carefully read the output specification. Submitted solutions are graded by a differential program, i.e., the output produced by a submission will be compared *byte-for-byte* with the output of the judges' solutions. Thus, even though submitted solutions may produce output that is *technically* correct, it will be counted as incorrect if these specifications are not followed exactly.

For all problems, solutions receive input from standard input (*stdin*) and produce output on standard output (*stdout*).

Restricting input and output to stdin and stdout does not prevent a program from processing files. Let `Main` be the program name, `Input` be the input file name, and `Output` be the output file name. The following command at the terminal allows the program to read and write the input and output files respectively:

```
./Main < Input > Output      # for C or C++, assuming Main is the executable
java Main < Input > Output   # for Java, assuming Main is a class file
```

For all problems, an *integer* is defined as an optional minus sign followed by one digit from the set 1..9 followed by zero or more digits from the set 0..9. `3`, `-21`, and `309` are integers; `0123`, `- 4`, and `+38` are not.

A *positive integer* is an integer whose value is greater than 0. You may assume the value of a positive integer is no more than $2^{31} - 1$, or $2,147,483,647$.

A *string*'s maximum length is 16,384.

For all problems, a *line* is defined as zero or more characters followed by a newline ('\n') character. Unless otherwise directed, a line of output should have no leading or trailing whitespace. A program producing the integer 27 on one line of output should consist of the character '2' followed by the character '7' followed by the newline ('\n') character.

Input for a problem consists of multiple test cases; each test case is followed by a blank line. You must read input until the *end-of-file*. Refer to the code for the Warm Up problem for hints and ideas for handling input properly.

For each test case, your program must output the result following the specified format in each problem. Output a blank line after the output of each test case. Refer to the code for the Warm Up problem again for hints and ideas for handling output properly.

1 Computing Social Security Retirement Benefit

"*The PIA is the benefit (before rounding down to next lower whole dollar) a person would receive if he/she elects to begin receiving retirement benefits at his/her normal retirement age.*"(www.ssa.gov) To compute a worker's Social Security Monthly Benefit Amount, we must first computer a worker's Primary Insurance Amount (PIA). The PIA is derived from a worker's Average Indexed Monthly Earning (AIME).

For folks reaching age 62 in 2017, a person's PIA is computed based on his/her AIME as follows. The 90% of the first $885 of his/her AIME, plus the 32% of his/her AIME amount over $885 and through $5336, and plus 15% his/her AIME over $5336. "We round this amount to the next lower multiple of $.10 if it is not alraedy a multiple of $0.10."(www.ssa.gov)

For example, if a worker's AIME is $5,000.00, his/her PIA is 885*0.9 + (5000-885)*0.32 = 796.50 + 1316.80 = 2113.30.

As another example, if a work's AIME is 10,0000, his/her PIA is 885*0.9 + (5336-885)*0.32 + (10,000 - 5336)*0.15 = 796.5 + 1424.32 + 699.6 = 2920.40. (2920.42 is rounded to 2920.40)

To make the computation easier, we will compute the PIA truncating down to the next lower whole dollar. Write a program to compute a person's PIA truncating down to the next lower whole dollar when his/her AIME is provided and assume the person reaches age 62 in 2017.

input Each test case has only one line containing an integer n for AIME where $1 \leq n \leq 10000$. *There is always a blank line after every test case.*

output For each test case, output two lines: the first line containing the corresponding PIA; *and a blank line as second*

Example:

Input

```
5000

10000
```

Output

```
2113

2920
```

2 Computing a worker's Average Indexed Monthly Earning and Social Security Retirement Benefit

We see how PIA is computed in the previous problem and the computation depends on Average Indexed Monthly Earning(AIME). A worker's AIME is determined using his/her entire working and earning history and the year in which he/she reaches age 62.

The amount of earning that is used to pay social security tax each year is called the nominal earning. To adjust earnings of the past to the present value, nominal earnings are adjusted to so called indexed earnings. The indexing or adjustment is done based on the year in which a person reaches age 62 and the National Average Wage Index computed each year by the Federal Government. The government uses previous year's National Average Wage Index multiplied by the percentage change in average wages (as measured by annual wage data) from the previous year to the current year to obtain the current year's National Average Wage Index.

If a person reaches 62 in 2017, the nominal earnings in 2016 and 2015 are not adjusted and they are the indexed earnings. For years earlier than 2015, the indexed earning of a year is obtained by multiply the nominal earning of that year by the ratio of the National Average Wage Index of 2015 and the National Average Wage Index of that year.

For example, if a worker's nominal earning in 2014 is 18000, the index earning is 18000 * (48098.63/46481.52) = 18626.40, where the National Average Wage Index of 2015 is 48,098.63 and the National Average Wage Index of 2014 is 46481.52.

The AIME is computed by first computing the index earning of each year, then adding the 35 highest index earnings (if an individual worked less than 35 years, we will add all the index earnings of the working years), then dividing the sum by 420, where 420=35x12 is the number of months in 35 years (as you see, one way to maximize your social security retirement benefit is to work for at least 35 years). The value obtained after the division is rounded down to the next lower dollar amount, which is the AIME.

Here is the National Average Wage Index of each year from the 1969 (the year a person was age 14 if he/she was born in 1955. Note age 14 is the minimum legal working age) to 2015.

```
1969 $5,893.76
1970 $6,186.24
1971 $6,497.08
1972 $7,133.80
1973 $7,580.16
1974 $8,030.76
1975 $8,630.92
1976 $9,226.48
1977 $9,779.44
1978 $10,556.03
1979 $11,479.46
1980 $12,513.46
1981 $13,773.10
1982 $14,531.34
1983 $15,239.24
1984 $16,135.07
1985 $16,822.51
1986 $17,321.82
1987 $18,426.51
1988 $19,334.04
1989 $20,099.55
1990 $21,027.98
1991 $21,811.60
1992 $22,935.42
```

```
1993 $23,132.67
1994 $23,753.53
1995 $24,705.66
1996 $25,913.90
1997 $27,426.00
1998 $28,861.44
1999 $30,469.84
2000 $32,154.82
2001 $32,921.92
2002 $33,252.09
2003 $34,064.95
2004 $35,648.55
2005 $36,952.94
2006 $38,651.41
2007 $40,405.48
2008 $41,334.97
2009 $40,711.61
2010 $41,673.83
2011 $42,979.61
2012 $44,321.67
2013 $44,888.16
2014 $46,481.52
2015 $48,098.63
```

For example, if a worker has the following earning history

```
1980 18000
1981 20000
1982 22000
2000 38000
2001 38000
2002 38000
2003 38500
2004 39000
2005 40000
2014 43000
2016 45000
```

The worker's AIME is $\{18000(48098.63/12513.46) + \ldots + 43000(48098.63/46,481.52) + 45000\}/420$, after rounding down to the whole dollar, 1494. The PIA is 991.30.

Note that the earning history in the example is less than 35 years. If it is more than 35 years, we need to pick the 35 highest indexed earning to computer the AIME.

Write a program to compute a person's AIME and PIA when his/her earning history from 1969 to 2016 is provided and assume the person reaches age 62 in 2017. Note that PIA should be computed including dollars and cents ("We round this amount to the next lower multiple of $.10 if it is not already a multiple of $0.10."(www.ssa.gov)). Also if the PIA is 1000 the output is 1000.00 and we always show the two digits after the decimal point in the output.

input Each test case contains no more than 49 lines. Each line contains two values separated by a space. The first value is the year and the second value is the nominal earning of that year in whole dollar amount. *There is always a blank line after every test case.* Input may contain multiple test cases.

output For each test case, print the corresponding AIME and PIA separated by a space *and a blank line.*

Example:

Input

```
1980 18000
1981 20000
1982 22000
2000 38000
2001 38000
2002 38000
2003 38500
2004 39000
2005 40000
2014 43000
2016 45000
```

Output

```
1494 991.30
```

3 Is the sum of 3 squares is a square ?

Given three positive integers x, y, z decide if the sum of squares of each number is a square of another integer, more precisely, we ask if $x^2 + y^2 + z^2 = k^2$ for some integer k. For example, $2^2 + 2^2 + 1^2 = 3^2$, which is a square of another integer(3), and $2^2 + 1^2 + 1^2 = 6$, which is not a square of any integer.

Write a program to decide if the sum of the squares of the 3 given positive integer is a square of some integer.

input Each test case has one line containing 3 positive integers separated by a space, where the value of each integer is no more than 200. *There is always a blank line after every test case.*

output For each test case, output two lines: the first line containing either `YES` or `NO`, where `YES` means the sum of the squares of the 3 integers is a square of some integer, and `NO` means otherwise; *and a blank line as second*

Example:

Input

```
2 2 1

2 1 1
```

Output

```
YES

NO
```

4 Can a square be expressed as the sum of three squares?

Given a square, decide if it may be expressed as the sum of the squares of three positive integers. For example, $9 = 3^2$ is a square and can be expressed as $9 = 2^2 + 2^2 + 1^2$ the sum of the squares of three positive integers. $16 = 4^2$ cannot be expressed as the sum of squares of three positive integers (you may want to verify that) Given an integer n, decide if n^2 may be expressed as the sum of three squares of some positive integers.

input Each test case has only one line containing an integer x where $1 \leq x \leq 2,000$. *There is always a blank line after every test case.*

output For each test case, output two lines: the first line containing either `YES` or `NO`, where `YES` means the square of the input is the sum of the squares of the 3 positive integers, and `NO` means otherwise; *and a blank line as second*

Example:

Input

```
3

4

5

6
```

Output

```
YES

NO

NO

YES
```

5 Computing stock buying signals and selling signals

The closing price of a stock is the last price traded for that security. For a given period, we may have a series of closing prices correspond to each day the market is open. A buying signal is used to determine if we should buy the stock on the next day. A selling signal is used to determine if we should sell the stock on the next day. The first sequence of three consecutive rising stock closing prices is a buying signal and we will buy the stock the next day if we have the money to invest. The first sequence of three consecutive rising stock closing prices is defined by considering 4 consecutive closing prices, P_1, P_2, P_3, P_4, where the following holds $P_1 > P_2$ and $P_2 < P_3 < P_4$. A special case is the first three closing prices in the data set is strictly increasing. Similarly, the first sequence of consecutive falling stock closing prices is a selling signal and we will sell the stock the next day if we own the stock. The first sequence of three consecutive falling stock closing prices is defined similarly.

For example, the following are the closing prices for facebook stock from 1-17-2017 to 2-14-2017.

```
127.87
127.92
127.55
127.04
128.93
129.37
131.48
132.77
132.18
130.98
130.32
133.23
130.84
130.98
132.06
131.84
134.20
134.14
134.19
134.05
133.85
```

The first sell signal is indicated by 127.92, 127.55, 127.04 and the first buy signal is indicated by 127.04, 128.93, 129.37. In this example, there are three selling and two buying signals. The other selling signals are 132.77, 132.18, 130.98 and 134.19, 134.05, 133.85. The other buying signal is 130.84, 130.98, 132.06.

Write a program to determine the number of buying signals and selling signals for a series of closing prices of a stock.

input Each test case contains no more than 1000 lines. Each line contains a number for the closing price. *There is always a blank line after every test case.* Input may contain multiple test cases.

output For each test case, print two integers in a line separated by one space *and a blank line.* The first number is the number of buying signals and the second number is the number of selling signals.

Example:

Input

```
127.87
127.92
127.55
127.04
128.93
129.37
131.48
132.77
132.18
130.98
130.32
133.23
130.84
130.98
132.06
131.84
134.20
134.14
134.19
134.05
133.85
```

Output

```
2 3
```

6 Automated Trading Strategy Simulation using Historical Data

To further study the buying signal and selling signal of the previous problem as a investment strategy, we will use the historical closing prices and perform simulations to determine gains or losses of the trading strategy.

Suppose we have $100,000.00 to invest. We will invest all the money to buy as many shares (number of shares is a whole number) as possible the next day after we see the first buy signal the previous day. We will use the average closing prices of the day we see a buying signal and the date we buy the stock as the stock buying price. We hold on to our investment until we see a selling signal. Then we will sell all our shares the next day. Again we will use the average closing prices of the day we see a selling signal and the date that we sell the stock as the stock selling price. We will repeat the buying process whenever we have cash on hand (either initially or sold all the shares) and selling process whenever we have shares in our hand. We will sell all the shares (if any) at the last closing price of the historical closing prices. The selling price is the last closing price. As you can see each round trip transactions (buying and selling) may results gains or losses. The overall gain or loss is the sum of all gains and losses.

Using the example of the previous problem, we buy 766 (obtained by dividing our cash on hand, which is 100000 at the moment, by 130.425) shares of facebook stock using the average prices of 129.37 and 131.48, which is 130.425. The cash left is 94.45. At the next sell signal, 766 shares are sold using the average price of 130.98 and 130.32, which is 130.65. As a result our cash on hand is 94.45 + 766 $\times$ 130.65, which is 100172.35. At the next buy signal, we use the average price of 132.06 and 131.84 (which is 131.95) to buy 759 shares and the cash left is 22.30. Since the last sell signal is at the end and there is no next day price to compute the average price in the example, which is the same as if there is no more sell signal, we sell all the shares using share price of 133.85 and obtain 759 $\times$ 133.85 + 22.3 total cash, which is 101614.45. So in this example, the gain is 1614.45.

Note that during the simulation, the cash amount remaining may have 3 digits after the decimal point or a digit after the penny digit. If this is the case, the 3rd digit should be rounded since the account balance is kept as dollars and cents.

Given a series of historical closing prices of a stock, determine the overall gain and loss of the trading strategy that is based on buying signals and selling signals as described above. Note that if the gain is 20.00 the output is 20 and the decimal point as well as the two digits after the decimal point are not show. Similarly if the gain is 15.20 the output is 15.2. If 15.20 is a loss the output is -15.2.

input Each test case contains no more than 1000 lines. Each line contains a number for the closing price. *There is always a blank line after every test case.* Input may contain multiple test cases.

output For each test case, print one number for gains or loses *and a blank line.* The gain or lose is represented by a positive or a negative value respectively.

Example:

Input

```
127.87
127.92
127.55
127.04
128.93
129.37
131.48
132.77
132.18
130.98
130.32
133.23
130.84
130.98
132.06
131.84
134.20
134.14
134.19
134.05
133.85
```

Output

```
1614.45
```

7 Sequence matching

Ribonucleic acid (RNA)consists of four "letters": A, U, G, and C. A RNA sequence is a sequence consisting of letters A, U, G, and C.

Given two RNA sequences, we want to identify the longest (common) subsequence that is a subsequence contained in both. For example, CAGCCUCGCUUAC and AAUGCCAUUGACGG have GCC as the longest common sequence and its length is 3.

Write a program to identify the length of the longest common subsequence of two RNA sequences.

input Each test case has two lines containing the two RNA sequences. The length of the each sequence is no more than 200. *There is always a blank line after every test case.*

output For each test case, output two lines: the first line containing the length of the longest common subsequence *and a blank line as second*

Example:

Input

```
CAGCCUCGCUUAC
AAUGCCAUUGACGG

A
C
```

Output

```
3

0
```

8 Sequence alignment

Two RNA sequences may be aligned against one another. There are many different alignments and different alignments may result in scores based on so called cost matrix. Consider the RNA sequences in the previous problem and the following shows a few alignments:

```
CA-GCC-UCGCUUAC--
AAUGCCAUUG---ACGG
```

```
CAGCCUCGCUUAC-
AAUGCCAUUGACGG
```

To compute the score the an alignment, we use the following score scheme. When two letters are the same in a position, score is 2. When two letters are different, score is 0. When a gap shows up, score is -0.5.

Hence, the score for the first alignment is 0+2+(-0.5)+2+2+2+(-0.5)+2+0+2+(-0.5)+(-0.5)+(-0.5)+2+2+(-0.5)+(-0.5)=12.5.

The score of the second alignment is 0+2+0+0+2+0+0+0+0+0+0+0+0+(-0.5)=3.5

Write a program to find the maximum score among all the possible alignments of two RNA sequences.

input Each test case has two lines containing the two RNA sequence. The length of the each sequence is no more than 200. *There is always a blank line after every test case.*

output For each test case, output two lines: the first line containing the maximum score *and a blank line as second.*

Example:

Input

```
CAGCCUCGCUUAC
AAUGCCAUUGACGG

A
C
```

Output

```
12.5

0
```