Taylor High School

Novice

Problem 2.1	**Welcome!**
General Statement	Introduce your teammates.
Time Allocation	1 second
Name of data file:	No input will be given.
Input	None
Output	Introduce yourself and your teammates in the following manner: `Knock Knock, Name, Name, and Name are trick-or-treating at your door.` Output will be formatted exactly like the sample output below.
Assumptions	No input will be given.
Sample Input	None
Sample Output	`Knock Knock, James, Ephraim, and Taylor are trick-or-treating at your door.`

Problem 2.2	**The Tattooist**
General Statement	Frank Li-enstein wants a fancy new tattoo on his arm; however, he wants to do it himself. The most elaborate tattoo he is able to draw is that of a lightning bolt. Print out a picture of what the lightning bolt will look like.
Time Allocation	1 second
Name of data file:	No input will be given.
Input	None
Output	A lightning bolt made of `*`s, shaped exactly like that of the sample output below.
Assumptions	No input will be given.
Sample Input	None
Sample Output	

Problem 2.3	**Inspiration**
General Statement	JimJin the Jack O' Lantern has decided to give out fortune cookies to trick-or-treaters instead of candy for Halloween. For each fortune cookie, she decides to write random words on the slip of paper that goes inside. Help JimJin think up of words to write inside the cookie.
Time Allocation	1 second
Name of data file:	No input will be given.
Input	None
Output	A word more than 6 characters long in all caps. Output should be formatted exactly like the sample output below.
Assumptions	No input will be given.
Sample Input	None
Sample Output	`EPITOMB`

Problem 2.4 **Unrated**

General Statement Vinnie the Vampire has given each house in his neighborhood a rating, depending on the amount of candy that the house has given in past years during Halloween. However, when Vinnie inputs the data into his computer, it began adding random negative signs in front of all the numbers! Help Vinnie write a program to determine whether the ratings are positive or negative so he can go trick-or-treating at the right houses.

Time Allocation 1 second

Name of data file: nov24.dat

Input The first line of input will be an integer *n* that determines the number of lines to follow. Each line will contain an arbitrary number of negative signs followed by an integer *x*.

Output Output should consist of a single word, either `positive` or `negative`. If the number of negative signs before a number is even, then it is positive. If it is odd, then it is negative.

Assumptions *n* will be an integer between 0 and 1,000,000 inclusive.
The number of negative signs will be greater than 0.
All input will be valid.

Sample Input

```
2
-----8
----4
```

Sample Output

```
negative
positive
```

Problem 2.5 **Pumpkin Range**

General Statement At a Halloween party, a game of Pumpkin Toss was played. The given list of numbers represent the distances in feet that the pumpkins were thrown (if it's negative, it represents a fail on the player's part). Output the range of the distances thrown.

Time Allocation 1 second

Name of data file: nov25.dat

Input The first line of input will be an integer *n* that determines the number of lines to follow. Each of the next *n* lines will contain the list of distances thrown in the game.

Output Output the range of the list of integers. The range of a list of numbers can be found by subtracting the smallest number in the given list from the largest number.

Assumptions *n* will be an integer between 1 and 1,000,000 inclusive.
The numbers in the list may be positive or negative.
The number of numbers in each list will be between 1 and 1,000,000 inclusive.
All input is valid.

Sample Input

```
8
4
5
7
8
1
-1
0
-10
```

Sample Output

```
18
```

Problem 2.6	**BOO!**
General Statement	Pheobe needs to measure a sheet of cloth so she can dress up as a ghost for Halloween. However, the measurements depend on her height. Use the formula below to output the dimensions of her sheet of cloth.
Time Allocation	1 second
Name of data file:	nov26.dat
Input	The first line of input will be an integer *n* representing the number of lines to follow. Each line will contain a single double *h*, which represents Pheobe's height in feet.
Output	Output the width and length of the sheet of cloth needed according to the formula below: Width = 1.25*h* Length = 2.5*h* Output should be in the format `W ft x L ft`, where `W` represents the width and `L` represents the length of the sheet.
Assumptions	*n* will be an integer between 0 and 1,000,000 inclusive. *h* will be a double between 0 and 1000 inclusive. All input will be valid.

Sample Input	`2`
	`4.25`
	`3.25`

Sample Output	`5.3125 ft x 10.625 ft` `4.0625 ft x 8.125 ft`

Problem 5.1 **Halloween Night**

General Statement: Halloween is here! Convey your excitement by outputting the date of this illustrious holiday in the special format displayed below.

Time Allocation: 1 second

Name of data file: There will be no data file.

Input: There will be no input.

Output: The output should be formatted exactly like the sample output below.

Assumptions: There is no input.

Sample Input: None

Sample Output:

```
HAL LOW EEN HAL L
O W E    E    N H
A L L    O  WEE N
H A L    L    O W
EEN HAL  L  OWE E
```

Problem 5.2 **Phantom of the Mall**

General Statement: Christiaan the Phantom needs to buy candy to hand out to trick-or-treaters on Halloween. However, after being dormant in his underground lair for so many centuries, he has forgotten simple addition and multiplication. He does know, though, that a large bag of candy costs $2.50, a medium sized bag costs $1.25, and a small bag costs $0.75. Help Christiaan figure out the total amount he must pay for the candy.

Time Allocation: 1 second

Name of data file: nov52.dat

Input: The first line of input will be an integer *n* indicating the number of lines to follow. Each line will have 3 integers, each separated by a single space, representing the number of large, medium, and small candy bags, respectively, that Christiaan buys.

Output: Output the total amount of money that Christiaan the Phantom spends, to the nearest penny. The output should be formatted exactly like the sample output below.

Assumptions: *n* will be an integer between 1 and 1,000,000 inclusive.
The number of bags of candy of each size will be between 0 and 100,000 inclusive.
All input will be valid.

Sample Input:

```
3
1 1 1
2 0 1
0 0 1
```

Sample Output:

```
$4.50
$5.75
$0.75
```

Problem 5.3 **Lost in Translation**

General Statement: Fin the Incorporeal received an e-mail from his friend, Beard the Bombastic about an upcoming Halloween party. However, Fin is having a hard time reading it, for he is a 19th century ghost of Ernest Rutherford and is not fluent in the language of emoticons, which Beard has used liberally throughout his e-mail. Help Fin translate the emoticons into plain English.

Time Allocation: 1 second

Name of data file: nov53.dat

Input: The first line of input will be an integer *n* representing the number of lines to follow. Each line of input will have a single emoticon, *e*.

Output: Translate the emoticon using the following definitions:

Emoticon	Definition
=D	I am happy
=P	I pwn you
=(	I am sad
<3	I like
=O	I am shocked
=3	I am smug

If *e* has a repetition of the second character (e.g. `=DDDDDD`), then for every repetition of a character *x* after the first appearance of *x*, an exclamation mark should be added to the end of the translation. Therefore `=DDDDDD` is equivalent to `I am happy!!!!!`

Assumptions: *n* will be an integer between 1 and 10,000 inclusive.
The number of characters in *e* will be between 2 and 1,000,000 inclusive.
All input will be valid.

Sample Input:
```
2
=PPPPPP
<3
```

Sample Output:
```
I pwn you!!!!!
I like
```

Problem 5.4 **House of Mirrors**

General Statement: Grumpy the Ghoul and Wendy the Witch are at the Halloween carnival, playing in the House of Mirrors. Grumpy cannot find Wendy, so he decides to write a message to her and reflect it in the multitude of mirrors so that she might see it no matter where she is. However, because the image is reflected, Wendy can only read Grumpy's message if the word is palindromic, (the same backwards and forwards), spaces removed, since she can read letters reflected in a mirror but not sentences. Help Grumpy determine whether or not Wendy can read his message.

Time Allocation: 1 second

Name of data file: nov54.dat

Input: The first line of input contains an integer *n* that represents the number of lines to follow. Each of the next *n* lines consists of a sentence *c* that Grumpy is considering.

Output: The output should consist of a single word `yes` or `no`, indicating whether or not the sentence *c* becomes a palindromic word after removing all spaces and converting every letter to lowercase.

Assumptions: *n* will be an integer between 1 and 1,000,000 inclusive.
c will contain only uppercase and lower case letters and spaces.
c will begin and end with a letter.
c will have between 1 and 1,000,000 characters inclusive.
All input will be valid.

Sample Input:

```
2
Sup not on a pus
Must sell at tallest sum
```

Sample Output:

```
no
yes
```

Problem 5.5 **The Invited**

General Statement: George the Gnome is a very lazy typist. While typing out his invitations for his Halloween party, he was too lazy to capitalize his letters and add correct punctuation. His mother, annoyed at the grammatical atrocities George has committed, wants to correct his mistakes. Help his mother write a program to correct the capitalization and punctuation mistakes in George's sentences.

Time Allocation: 1 second

Name of data file: nov55.dat

Input: The first line of input contains an integer *n* that represents the number of lines to follow. The next *n* lines each have a single sentence *s*. *s* may contain any number of words.

Output: The output should consist of the sentence *s* with the correct punctuation and capitalization, according to the rules below:

1) The first letter of the first word of each sentence should be capitalized.
2) If adjacent words are separated by two spaces, a comma must be inserted and the number of spaces between the two words reduced to one. (Therefore `Hi  Bob` becomes `Hi, Bob`).
3) All sentences must end in a period and only a period.
4) The word "`I`" should always be capitalized.

Assumptions: *n* will be an integer between 1 and 1,000,000 inclusive.
s will contain only words and spaces.
S may or may not contain uppercase letters.
s will have between 1 and 1,000,000 characters inclusive.
There will be no more than two spaces between adjacent words.
All input will be valid.

Sample Input:

```
2
you  mister vampire are invited to my party
do not forget i love sugar  so bring candy
```

Sample Output:

```
You, mister vampire are invited to my party.
Do not forget I love sugar, so bring candy.
```

Problem 5.6 **Sharing is Caring**

General Statement: Bob is very excited to go trick-or-treating. Unfortunately, his bully of a brother, Hubert, insists that he share his candy with him once Bob has finished, since Hubert is too lazy to go trick-or-treating himself. Unwilling to do so, Bob strikes a deal: he will only share his candy with Hubert if the number of pieces he obtains is a power of 2. Help Bob figure out whether the number of pieces of candy he has is an integral power of 2.

Time Allocation: 1 second

Name of data file: nov56.dat

Input: The first line of input contains an integer *n* that represents the number of lines to follow. The next *n* lines will have a single integer *p*, signifying the number of pieces of candy Bob has.

Output: The output should consist of a single lowercase word, either `yes` or `no`, indicating whether *p* is an integral power of two. The output is to be formatted exactly like the sample output given below.

Assumptions: *p* will be a positive base ten number between 1 and 1,000,000 inclusive.
n will be an integer between 1 and 1,000,000 inclusive.
An integral power of two is defined as any integer *y* of the form 2^x, where *x* is an integer greater than or equal to 0.
All input will be valid.

Sample Input:
```
2
2
10
```

Sample Output:
```
yes
no
```

Problem 9.1 **Paranoia 0.0**

General Statement: Nefis the Zombie is a very paranoid zombie. When decorating her windows for Halloween, she insists that all designs must be vertically symmetrical. However, her eyesight is not very good, due to a mishap in one of her previous experiments. Help Nefis determine if her decoration is symmetrical or not.

Time Allocation: 1 second

Name of data file: nov91.dat

Input: The first line of input contains an integer *n* that represents the number of data collections to follow. The first line of each data collection consists of two integers *l* and *w*, indicating the length and width of Nefis' rectangular window. The next *l* lines will contain the design on the witch's window, which consist only of capital letters.

Output: The output should consist of a single word `yes` or `no`, indicating whether or not the design on Nefis' window is symmetrical. A design is considered symmetrical if all the characters on its vertical axis (the column that is in the exact middle of the design) are the same and the design can be reproduced by reflecting the characters on the left or right side of the axis to the opposite side. Therefore,

```
BCXCB
ACXCA
BCXCB
```

is symmetrical, while

```
 QOQ
 TOP
 POP
```

is not. Output should be formatted exactly like the sample output below.

Assumptions: *n* will be an integer between 1 and 1,000,000 inclusive.
l will be an integer between 1 and 1000 inclusive.
w will be an odd integer between 1 and 999 inclusive.
The design will consist of only capital letters.
All input will be valid.

Sample Input:

```
2
3 3
AJA
IJO
FOF
4 5
ARLRA
WOLOW
RKLKR
CJLJC
```

Sample Output:

```
no
yes
```

Problem 9.2 **Final Destination**

General Statement: Nory the Ninja is participating in a Halloween treasure hunt around her neighborhood, and since the prize is an inordinate amount of candy, she is extremely motivated to win. The treasure hunt consists of a series of clues that lead to the final goal of candy heaven. Each clue leads to a place where there are multiple clues – the idea is that each contestant must test out every clue, for all but one leads to a dead end. However, Nory, being a ninja, can use her ninja skills to determine which clue is correct with the following method: At each clue location there will be a specific "key word" unbeknownst to the other contestants. If all the letters of a clue appear in order within that key word, then it is the correct clue. For example, `abd` is a correct clue for the key word `abcd`, but `cde` and `bca` are not. Help Nory determine whether or not a clue is correct.

Time Allocation: 1 second

Name of data file: nov92.dat

Input: The first line of input will be an integer *n* representing the number of data collections to follow. Each data collection will consist of a key word *s* followed by a clue in consideration, *t*. *s* and *t* will be on separate lines.

Output: The output should consist of a single lowercase word, either `yes` or `no`, indicating whether *t* is a correct clue or not. The output is to be formatted exactly like the sample output given below.

Assumptions: *s* and *t* will contain between 1 and 1000 characters, inclusive.
s will contain only lowercase letters and spaces.
t will contain only lowercase letters (no spaces).
s and *t* will each begin and end with a letter.
All input will be valid.

Sample Input:

```
2
boulevard
board
jack o lantern
cooler
```

Sample Output:

```
yes
no
```

Problem 9.3 **Candy Stacking**

General Statement: Will the Werewolf has bought large amounts of candy in anticipation of the trick-or-treaters that come visit him on Halloween night. Bored while waiting for his doorbell to ring, he decides to organize his candy in a peculiar manner. First, he takes an arbitrary number of buckets and stacks them up in a pyramid formation, with one more bucket in each successive row (going downwards). Then, he fills the buckets with candy according to the Fibonacci series. If the buckets are numbered and stacked like so:

```
1
2 3
4 5 6
7 8 9 10
…
```

, then the first bucket will have 1 piece of candy, the 2nd will have 2 pieces, the 3rd will have 3 pieces, the 4th will have 5 pieces, and so on, continuing with the Fibonacci sequence until the required number of buckets are filled. Help Will figure out how many pieces of candy he must put in each bucket.

Time Allocation: 1 second

Name of data file: nov93.dat

Input: The first line of input will be an integer *n* representing the number of lines to follow. Each line consists of a single integer *x* that represents the number of buckets to be filled with candy.

Output: The output should consist of Fibonacci numbers in a pyramid formation, representing the amount of candy in each bucket, which are stacked like a pyramid. If there are not enough buckets to make a perfect pyramid, then the pyramid should be constructed from the top downwards until there are no more buckets (yes this defies gravity). Each row is filled from left to right. A single space should separate each number in the pyramid. A blank line should separate each pyramid. The output should be formatted exactly like the sample output below.

Assumptions: *x* will be an integer between 1 and 90 inclusive.
(Warning – the *nth* Fibonacci number for *n*>45 is greater than 2^{32})
Trailing spaces at the end of rows will not be judged.
All input will be valid.

Sample Input:

```
2
8
15
```

Sample Output:

```
1
2 3
5 8 13
21 34

1
2 3
5 8 13
21 34 55 89
144 233 377 610 987
```

Problem 9.4 **Anaconda Doodling**

General Statement: Stewart the Studious Student is an unfortunate soul and has to take classes on the night of Halloween, so while dreaming earnestly about candy and costumes, he takes notes in a very strange manner. Every sentence he writes is in a snake-like formation (the concentration needed to do this keeps him focused). Given a sentence that his teacher says, output what it will look like in his notes.

Time Allocation: 1 second

Name of data file: nov94.dat

Input: The first line of input will be an integer *n* representing the number of data collections to follow. Each data collection will consist of two lines: the first line of input will be an integer *x*, representing the width of the snake. The next line of input will be a string *s*, which is to be reformatted and output.

Output: The output should consist of the string *s* output in the snake like formation, left-justified, with a width *x,* like those demonstrated in the sample output. A word snake is created by first writing *x* letters from left to right, 1 letter directly below the last letter, then x letters from right to left, then 1 letter directly below the last letter, and so on, making a snake like shape. There will be no spaces or punctuation in the word snake, and all letters will be capitalized. Each output should be separated by a line of space.

Assumptions: *n* will be an integer between 1 and 1,000,000 inclusive.
s may contain punctuation, spaces, and upper and lower case letters.
Punctuation in *s* is limited to commas and ending punctuation.
s will have between 2 and 100,000 characters inclusive.
x will be an integer between 2 and 100 inclusive.
The output must contain only uppercase letters.
All input will be valid.

Sample Input:

```
2
3
Pumpkins
5
Frankenstein is crashing the party!
```

Sample Output:

```
PUM
  P
NIK
S

FRANK
    E
IETSN
N
ISCRA
    S
TGNIH
H
EPART
    Y
```

Problem 9.5 **Lucky Lucky!**

General Statement: Casper decided he wants to dress up as a basketball player for Halloween. He tries to think of a number to put on the back of his jersey, and decides to put a random lucky number on his back to give him just that, luck. A lucky number in mathematics is a natural number obtained using the following method:
If you take the list of integers, beginning with 1:

{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21…}

Look at the first number after 1. Since 2 is the first number after 1, then you must cross out every second number in the set:

{1, ~~2~~, 3, ~~4~~, 5, ~~6~~, 7, ~~8~~, 9, ~~10~~, 11, ~~12~~, 13, ~~14~~, 15 ~~16~~, 17, ~~18~~, 19, ~~20~~, 21…}

Since 2 was the last number used, find the next number (not crossed out) after 2. Since that number is 3, cross out every third number remaining (do not count the numbers already crossed out).

{1, ~~2~~, 3, ~~4~~, ~~5~~, ~~6~~, 7, ~~8~~, 9, ~~10~~, ~~11~~, ~~12~~, 13, ~~14~~, 15 ~~16~~, ~~17~~, ~~18~~, 19, ~~20~~, 21…}

And repeat the process over and over again. The number *c* that determines which numbers are to be crossed off is the next number after the previous *c* that was used. For example, the next *c* in the example above would be 7, since it is the next number after 3, the number that was previously used. The numbers that are left after crossing numbers out from the list of integers are called lucky numbers. (Since they are lucky enough not to be crossed off) Help Casper find the *x*th lucky number to put on the back of his jersey.

Time Allocation: 1 second

Name of data file: nov95.dat

Input: The first line of input contains an integer *n* that represents the number of lines to follow. On each of the next *n* lines is an integer *x,* representing the *x*th lucky number you must output. Output should be formatted exactly like the sample output below.

Output: The *x*th lucky number.

Assumptions: *x* will be an integer between 1 and 1000 inclusive.
All input will be valid.

Sample Input:

```
2
3
5
```

Sample Output:

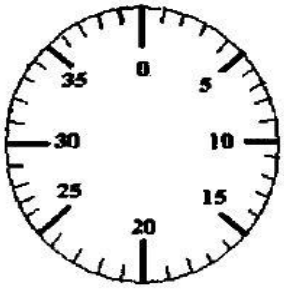
```
7
13
```

Problem 9.6 **Just the Right Combination**

General Statement: Finch the Dwarf, being paranoid, has locked up his share of candy obtained from his hard earned trick-or-treating in his trusty safe. Unfortunately, the next day, he forgot his 3-number combination, and he is devastated, for he really wants to eat his candy. Luckily, Finch remembers a trick to figuring out the correct locker combination by determining for which numbers the lock "clicks" when turned past those numbers. When he tries this, Finch discovers that four numbers click when he turns it clockwise, three numbers click when he turns it counterclockwise afterwards, and then only 2 numbers click when he turns it clockwise a final time.

Finch soon remembers some other rules. If a locker combination is represented as `A-B-C`, where `A, B, C` represent positive integers, then the 2nd number must be a minimum of 180º clockwise away from the 1st number, and the 3rd number must be a minimum of 90º counterclockwise and maximum of 180º counterclockwise away from the 2nd number. Also, the sum of `A, B,` and `C` cannot exceed 70.

If his lock looks like so:



with only the numbers 0 through 39 on it, help Finch determine which of the possible combinations is the right one. There are 9° between each tick on the lock. For example, it is 45 degrees from the digits 0 to 5 on the lock above.

Time Allocation: 1 second

Name of data file: nov96.dat

Input: The first line of input will be an integer *n* representing the number of data collections to follow. Each data collection consists of three lines: the first line will contain 4 integers, each separated by a single space, representing the possible numbers for `A`, the second line will contain 3 integers, each separated by a single space, representing the possible numbers for `B`, and the third line will contain 2 integers, each separated by a single space, representing the possible numbers for `C`.

Output: The correct combination of Finch's lock in the form `A-B-C`, where `A, B,` and `C` represent positive integers between 0 and 39 inclusive. Output should be formatted like the sample output below.

Assumptions: *n* will be a number between 1 and 100,000 inclusive.
Every number in each data collection will be between 0 and 40, exclusive.
There will be only one valid combination for each input.
All input will be valid.

Sample Input:

```
2
1 20 15 31
20 36 31
25 27
3 13 23 33
32 1 2
15 10
```

Sample Output:

```
1-36-25
3-32-15
```