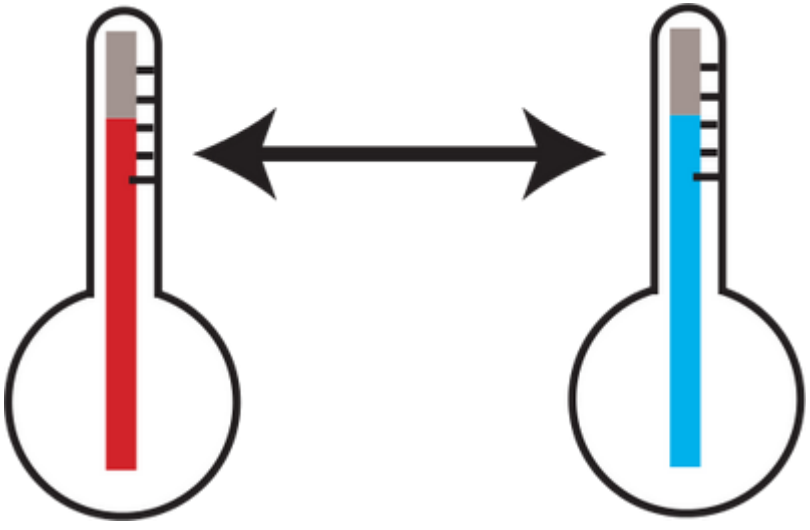
1. Temperature Converter ★

Your task is to write a temperature converter. The converter has two functions: convert from celsius to fahrenheit and convert from fahrenheit to celsius. The user is able to select between these functions in a main menu.

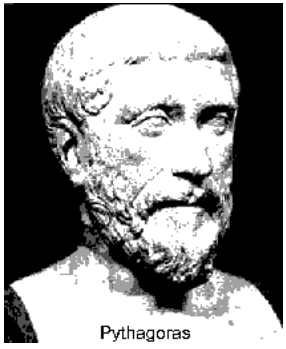


If the user chooses to convert from celsius to farenheit, the program should ask for the temperature in celsius and then output the equivalent temperature in fahrenheit. If the user chooses to convert from farenheit to celsius, the program should ask for the temperature in fahrenheit and then output the corresponding temperature in celsius.

After displaying the temperature, the program should go back to the main menu until the user chooses to exit.

2. Perfect Numbers ★

The discovery of perfect numbers was lost in prehistory, but it was known that the Pythagoreans (founded 525 B.C.) studied them for their mystical properties. The mystical tradition was continued by the Neo-Pythagorean philosopher Nicomachus of Gerasa. He gave moral qualities to his definitions, and such ideas found credence among early Christian theologians. Often the 28-day cycle of the Moon around the Earth was given as an example of a "Heavenly," hence perfect, event that naturally was a perfect number.


Pythagoras

A perfect number is a positive integer which is equal to the sum of its proper divisors. A proper divisor means all of its divisors except itself. The first perfect number is 6, which has the proper divisors 1, 2, and 3.

$$6 = 1 + 2 + 3$$

The next perfect number is 28, with proper divisors 1, 2, 4, 7, and 14.

$$28 = 1 + 2 + 4 + 7 + 14$$

While most people do not believe in the mystical properties of these numbers today anymore, perfect numbers are still studied in mathematics because they are cool. Write a program that accepts a positive integer n. The program should then output if n is a perfect number or not.

3. Addicted to X ★★

You are a programmer who is addicted to X (the letter). There are a lot of nice things that start with the letter X, such as X-ray, xylophone, X-Men and Xbox. Because of this addiction, you want to write programs that have a lot of X (just for kicks). This is one such program.

Write a program that accepts a positive odd integer n. The program should output an X figure (made of X's, of course) of size n. Refer to the examples below:

$n = 3$

```
X X
 X
X X
```

$n = 5$

```
X   X
 X X
  X
 X X
X   X
```

$n = 7$

```
X     X
 X   X
  X X
   X
  X X
 X   X
X     X
```

4. Crack the Code ★★

In computer security, passwords are not usually stored as is in the database. The reason for this is when the information from the database gets stolen, the passwords should be encrypted in some way so that the thief cannot easily figure out the correct passwords. Encryption refers to transforming a certain piece of data into a secret code, which can only be understood if you know the algorithm used to encrypt the data. There are a lot of complex encryption algorithms and researchers have really been putting a lot of effort to make encryption schemes which are difficult to break.

For this problem we will only be considering a very simple encryption scheme. We will assume that the password only contains digits, and it will not exceed 10 characters in length. To encrypt the password, we take each digit n and replace it with $(n + 5) mod 9 + 1$. For exampe, if we have the password:

$$12345$$

To encrypt it we will take each digit n and replace it with $(n + 5) mod 10$. Our encrypted password is:

$$78912$$

Your task is to write a program that, given a password **that has been encrypted in this manner**, will decrypt it and display the original password. For example, if the input is 78912, the answer should be 12345.

5. Date Validator ★★

Your task is to write a program that will ask for date. The date is composed of three parts: the month (represented as an integer where 1 is January, 2 is February, and so on), the date (a positive integer), and the year (another positive integer). The program should determine if the date is valid or not. If it is valid, output `You have entered a valid date`. If it is not valid, the program should ask the user to try again until he has inputted a valid date.

A valid date is described as follows:

(1) The month should obviously be in the range of 1 – 12.

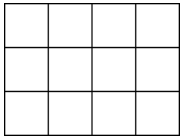
(2) January, March, May, July, August, October and December have 31 days. April, June, September, and November have 30.

(3) February has 28 days, except when it's a leap year when February has 29 days.

(4) A leap year is a year divisible by 4 but not divisible by 100, with exception of years divisible by 400.

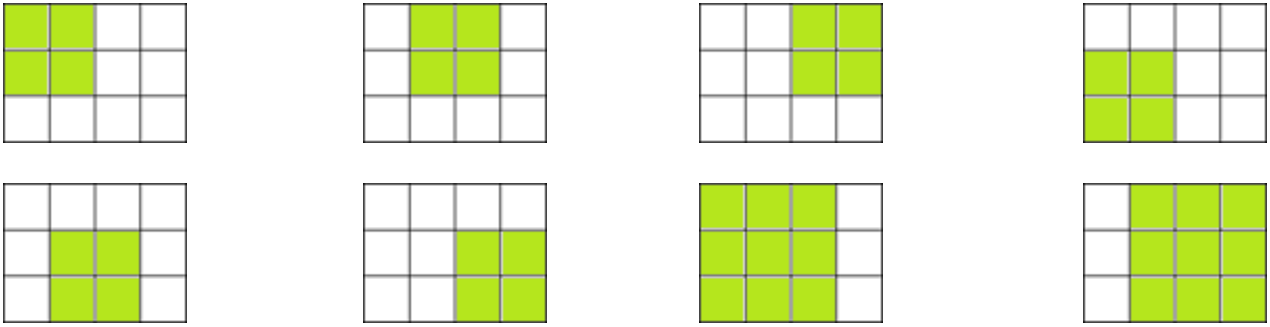
6. Bathroom Tiles ★★★

Many bathrooms are covered with grids of square tiles. The figure below shows a 3×4 grid of tiles (3 rows and 4 columns).



Your task is to write a program that will output the total number of squares in the grid of tiles. For a 3×4 grid of tiles as shown above, the output should be 20.

Yes, you've read that right. The answer is **20**, and not 12. As you can see, in addition to the 12 individual squares, there are 8 other squares that span more than one tile, as shown below:

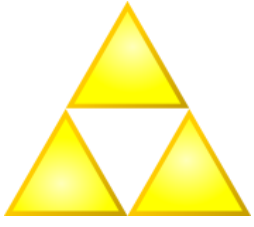


Your program should ask for the dimensions of the grid (number of rows and number of columns). Your program should then output the total number of squares that can be found in the resulting grid.

To help you, here are a few more examples. A 2×2 grid has a total of 5 squares (4 individual squares and one large square). A 1×5 grid has 5 squares (there are no larger squares aside from the individual ones). A 5×6 grid has a total of 70 squares.

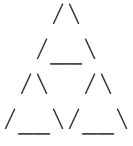
7. The Legend of Zelda ★★★

The Legend of Zelda is one of the most popular video game franchises of Nintendo. The game centers on the adventures of Link, the hero of the kingdom of Hyrule. One of the most iconic symbols of the game is the Triforce. The Triforce is a sacred relic able to grant the wishes of whoever touches it. It is composed of three equally-sized triangles, arranged in a certain fashion as shown below.

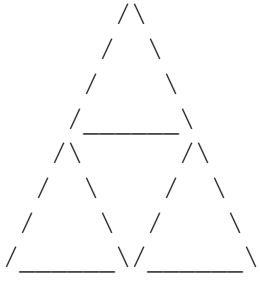


Your task is to write a program that asks for the size of the Triforce t. The program should then display a Triforce wherein each triangle is of size t. Refer to the examples below. ***t* can never be less than 2.**

$n = 2$



$n = 4$



$n = 5$