Programming Contest for Girls

Example Problems v2.1 ANSWERS

Contents

The Moon

Problem 1 (5 points)

Print, as a single integer on a single line, the total number of moons in orbit around the third planet from the sun.

Explaination

The third planet from the sun is Earth, which has a single moon. The answer is 1.

You are **not** allowed access to the internet or AI during the contest. If you do not know which planet is the third from the sun, it is a valid strategy to submit guesses until you find the solution. There is no limit to the number of submissions you can make during the contest.

Example Solution

```
print(1)
```

Multi-Pie

Problem 2 (5 points)

You work at a bakery making pies. Pies are arranged in rows on a large baking tray. Given the number of pies that fit in each row and the number of rows that fit on each tray, how many pies can you fit on a tray? As input, you will receive two positive **integers** each on their own line, representing the pies per row and the number or rows, respectively. Print the total number of pies you can fit on a tray.

Input

```
3
5
```

Output

```
15
```

Input

```
4
7
```

Output

```
28
```

Explaination

If a pies fit on a row, then $a \times b$ pies fit on b rows. Multiply the two numbers.

Example Solution

```
a = int(input())
b = int(input())
print(a * b)
```

Tech Support

Problem 3 (5 points)

Your friend is always asking for help using her phone. She asks the same questions so frequently that you've decided to write a program to automate your replies. As input, you will receive two space-separated strings on a single line, representing the the words to insert into your reply. Print the phrase "`Don't worry! Go to X and then find Y.`", where `X` is replaced with the first given string and `Y` is replaced with the second given string.

Input

```
Settings Bluetooth
```

Output

```
Don't worry! Go to Settings and then find Bluetooth.
```

Explaination

Use `.split()` to separate the first line of input into two strings. Substitute these two strings into the output. Make sure there is no space between the full stop and the second string. There are multiple valid ways to do this. E.g. an f-string, string concatenation with `+`, and using `print` with `end=` or `sep=`.

Example Solution 1

```
line = input()
a, b = line.split()
print(f"Don't worry! Go to {a} and then find {b}.")
```

Example Solution 2

```
a, b = input().split()
print("Don't worry! Go to " + a + " and then find " + b + ".")
```

Example Solution 3

```
a, b = input().split()
print("Don't worry! Go to ", a, " and then find ", b, ".", sep="")
```

Example Solution 4

```
a, b = input().split()
print("Don't worry! Go to", a, "and then find", b, end="")
print(".")
```

Ribbons

Problem 4 (5 points)

You and your friend are making decorations for a birthday party using ribbon. Write a program that calculates how many decorations you can make with a given length of ribbon.

The first line of input is a single non-negative integer, the length of ribbon you have. The next line contains the length, as a non-negative integer, of ribbon you need to make a single decoration.

Output the number of decorations you can make with the ribbon you have on a single line. On the next line, output the amount of ribbon you will have left over.

Input

```
5
3
```

Output

```
1 2
```

Input

```
30
10
```

Output

```
3 0
```

Explaination

In Python you can do an **integer division** using `a // b`, which means divide `a` by `b` and round down to the nearest integer. Equivalently, how many whole `b` go into `a`.

You can get the **remainder** using `a % b`, which means what is left over after `a // b`.

Example Solution

```
a = int(input())
b = int(input())
print(a // b, a % b)
```

Party Hats

Problem 5 (15 points)

Write a program to draw a party hat of a given size. A party hat forms an equilateral triangle with a given number of `*` on each side. There is a single space between each star on the last row. There is no leading space on the last row. A hat is symmetric about its vertical centreline. A hat of size 1 is just a single star. Input is a single positive integer (greater than zero) on a single line, the number of asterisks along each side. Your output must match the expected output exactly.

Input

```
3
```

Output

```
  *
 * *
* * *
```

Input

```
4
```

Output

```
   *
  * *
 *   *
* * * *
```

Explaination

A helpful technique with any problem is to break it down into achievable steps.

For example:

1. Start by writing a program that prints the bottom row of the triangle. Eg. `* * *` for an input of `3`.
2. Add the top row: a single star that is centred using space characters `" "`. Can we calculate how many spaces we need for a given size? Can we then construct a string of this many spaces and print it?
3. Each middle row contains two stars. Try printing just the left side of the triangle.
4. Finally, add the right side into the middle rows.

You will also need to think about the case when the size is 1. In this case, the answer is a single star. One way is to only print the bottom row when `size > 1`.

Example solutions are on the next page…

Example Solution 1

```
size = int(input())

# Top row
top = ""
for i in range(size - 1):
    top = top + " "
print(top + "*")

# Middle rows
for row in range(1, size - 1):

    # Leading spaces
    mid = ""
    for i in range(size - row - 1):
        mid = mid + " "

    # First star
    mid =  mid + "*"

    # Middle spaces
    for i in range(2 * row - 1):
        mid = mid + " "

    # Last star
    print(mid + "*")

# Bottom row
if size > 1:
    bot = ""
    for i in range(size):
        bot = bot + "* "
    print(bot)
```

Note that it is okay to print extra spaces at the end of a line. These will be ignored when comparing your answer to the correct answer.

Example Solution 2

```
size = int(input())
print(" " * (size - 1) + "*")
for i in range(1, size - 1):
    print(" " * (size - i - 1) + "*" + " " * (2 * i - 1) + "*")
if size > 1:
    print("* " * (size - 1) + "*")
```

This uses string multiplication to do the same as the previous solution. There are often multiple correct ways to solve a problem.