Assignment 2 (Draft)

Instructions

- Unless specified otherwise,
  - you can use anything in the Python Standard Library;
  - don't use any third-party library in your code for this assignment;
  - you can assume all user inputs are always valid and require no validation.
- The underlined blue texts in the Sample Runs section of each question refers to the user inputs. It does NOT mean that your program needs to underline them.
- Please follow closely the format of the sample output for each question. Your program should produce **exactly** the same output as the sample (same text, symbols, letter case, spacing, etc.). The output for each question should end with a single newline character, which has provided by the `print()` function by default.
- If you see fit to define your own functions for specific tasks in any question, you may do so, but this is not required.
- Name your script files as q1.py for Question 1, q2.py for Question 2, …, q5.py for Question 5 (case-sensitive). Using other names may affect our marking and may result in deduction.
- Your source files will be tested in script mode rather than interactive mode.

Question 1 – ASCII Art (??%)

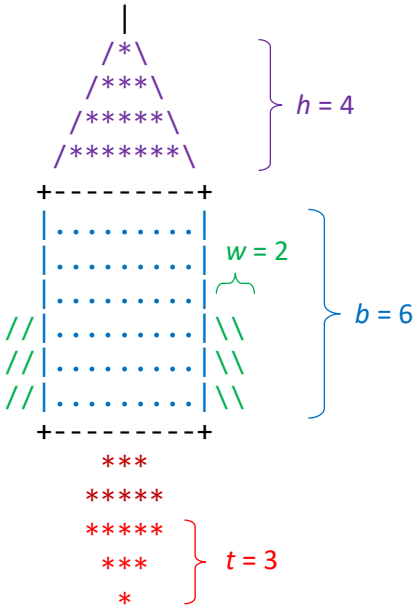
You are to write a program to print a rocket pattern using symbols in the ASCII character set as console output such as the following:

```
      |
     /*\
    /***\
   /*****\
  /*******\
 +---------+
 |.........|
 |.........|
 |.........|
//|.........|\\
//|.........|\\
//|.........|\\
 +---------+
    ***
   *****
   *****
    ***
     *
```

There are four parameters to control the overall shape of the rocket:

1. ***h* (head)**: controls the height of the rocket's head.
2. ***b* (body)**: controls the height of the rocket's body.
3. ***w* (wing)**: controls the width of the rocket's wings.
4. ***t* (tail)**: controls the height of the rocket's tail (or flame).

To ease visualizing which part of the shape each of these parameters refers to, please look at the following drawing with each parameter annotated:



Part 1: Rocket Head

For the first line to print, the rocket pattern begins with a single pipe symbol "|", which acts as the tip of the rocket's head.

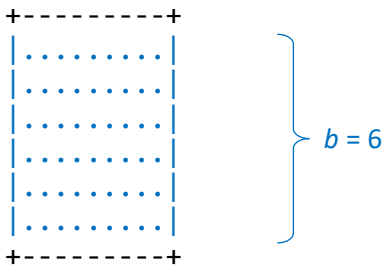
Then the program prints a triangular pattern which has its left and right edges formed by forward slashes "/" and backslashes "\" respectively, and is filled with asterisks "*". For example, it looks like:

```
   /*\
  /***\
 /*****\
/*******\
```

The number of "*" per line is always an odd number. The height of this triangular shape (a.k.a. the height of the head) is denoted by *h*, which is entered by the user via standard input. The above example refers to the case of *h* = 4, so this triangular shape is composed of four lines. The height of the head refers to the number of lines of asterisks "*" only, excluding the "|" tip and the line that follows, which looks like "+---------+".

Part 2: Rocket Body

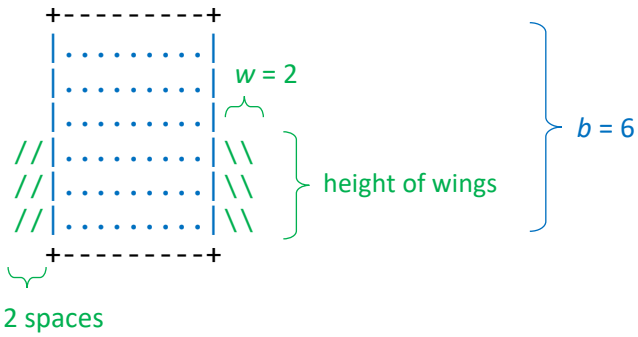
There are two lines "+---------+" at the top and bottom of the rocket's main body. They are formed by hyphens and a plus at the start and end positions of the line ("+" 's are printed at the four corners of the rocket's body).



The body part is filled with dot "."; the left and right edges are formed by the pipe "|" symbol.
Note that *b* refers to the height of the body; it counts the lines of dots only, excluding the top and bottom lines "+---------+". So, for the above example, *b* is 6 only.

Part 3: Rocket Wings

The rocket has two wings: the left wing and right wing are denoted by patterns of forward slashes "/" and backslashes "\" respectively.

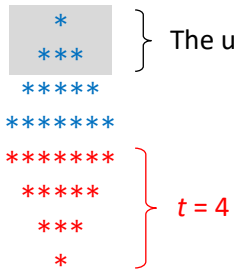


The two wings always have equal widths and heights. The height of the wings is set to half of the rocket body's height. For indivisible cases, we take the ceiling of the division, i.e., $\left\lceil \frac{b}{2} \right\rceil$.

For the above example, the wings have a width of 2, i.e., *w* = 2, and a height of 6 / 2 = 3.
Let's say if *b* = 7, then 7 / 2 = 3.5, taking the ceiling gives 4, so the wings' height is also 4.

Part 4: Rocket Tail (Flame)

The tail or flame part is the most complicated part in this assignment. It looks like a diamond shape but with some subtle differences. It is formed in the following way. Think of it as if two triangles are stacking on top of another, with the bottom one inverted:



What's more, the upper half of the top triangle is chopped (grey out in the above diagram).
The parameter *t* refers to the height of the bottom triangle. For the above example, *t* = 4.
For odd *t*, the number of lines of asterisks to keep for the top triangle is the ceiling of the division of *t* by 2, i.e., $\left\lceil \frac{t}{2} \right\rceil$.

The following table shows the shapes of the tails for different values of *t*. For example, for *t* = 5, we have 5 / 2 = 2.5, taking the ceiling gives 3. So, we keep/print the lower 3 lines and chop the upper 2 lines from the top triangle for the case of *t* = 5.

t	2	3	4	5	6	7	8
Shape of the tail	*** *** *	*** ***** ***** *** *	***** ******* ******* ***** *** *	***** ******* ********* ********* ******* ***** *** *	******* ********* ********** ********** ********* ******* ***** *** *	******* ********* ********** ********** ********** ********* ******* ***** *** *	********* ********** ********** ********** ********** ********** ********* ******* ***** *** *

Requirements on Input Validation

For the drawing to look like a rocket, the following bounds on the input values are enforced:

- $h \geq 1$
- $b \geq 2$
- $w \geq 1$
- $t \geq 2$

There is no upper bound validation necessary for the parameters *h*, *b* and *w*. But due to the visibility constraints of a console screen, we assume only small values for *h*, *b* and *w* for testing your program, practically up to around 30 for each of these parameters.

However, for *t*, its size should have an upper bound since we don't want the tail to look wider than the rocket body. Let's do some analysis below:

- The width (number of "*") of the base of the triangle for the head part is $2h - 1$.
- The width (including all characters "|.....|") of the body is $2h + 3$.

- The width of the tail (flame), i.e., $2t - 1$, should NOT be wider than the width of the rocket body. So, the user inputs must satisfy the following inequality:

$$2t - 1 \leq 2h + 3$$

You may assume that the user always enters positive integer values. However, if the user inputs for the 4 parameters violate any of the above requirements, your program will prompt the user again until a valid set of values is received.

Sample runs

```
Enter h, b, w, t: 1 2 1 2↵
   |
  /*\
 +---+
 |...|
/|...|\
 +---+
  ***
  ***
   *
```

```
Enter h, b, w, t: 2 2 3 3↵
      |
     /*\
    /***\
   +-----+
   |.....|
///|.....|\\\
   +-----+
     ***
    *****
    *****
     ***
      *
```

```
Enter h, b, w, t: 4 6 2 3↵
       |
      /*\
     /***\
    /*****\
   /*******\
  +---------+
  |.........|
  |.........|
  |.........|
//|.........|\\
//|.........|\\
//|.........|\\
  +---------+
     ***
    *****
    *****
     ***
      *
```

```
Enter h, b, w, t: 5 5 4 7↲
          |
         /*\
        /***\
       /*****\
      /*******\
     /*********\
    +-----------+
    |...........|
    |...........|
////|...........|\\\\
////|...........|\\\\
////|...........|\\\\
    +-----------+
       *******
      *********
     ***********
    *************
    *************
     ***********
      *********
       *******
        *****
         ***
          *
```

```
Enter h, b, w, t: 1 1 1 1↲      ← b and t are too small!
Invalid input!
Enter h, b, w, t: 1 2 1 1↲      ← t is too small!
Invalid input!
Enter h, b, w, t: 1 1 2 2↲      ← b is too small!
Invalid input!
Enter h, b, w, t: 3 4 5 8↲      ← t is too big!
Invalid input!
Enter h, b, w, t: 3 4 5 6↲      ← t is too big!
Invalid input!
Enter h, b, w, t: 3 4 5 5↲
          |
         /*\
        /***\
       /*****\
      +-------+
      |.......|
      |.......|
/////|.......|\\\\\
/////|.......|\\\\\
      +-------+
        *****
       *******
      *********
      *********
       *******
        *****
         ***
          *
```

Question 2 – Sudoku Checker (??%)

This question is about implementing a working Sudoku checker. Recap that a Sudoku is a 9x9 grid that is completed when every 3x3 square, row and column consist of the numbers 1-9.

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

Write a function `is_sudoku()` that receives a completed 9x9 grid, in the form of a two-dimensional list, and returns True if the grid represents a valid Sudoku and False otherwise. You may define other functions as callees for this function if it can help decompose the overall task better.

Note: we will import your module into our test scripts for testing against our test cases such as the following below:

```
A = [
    [4,3,5,2,6,9,7,8,1],
    [6,8,2,5,7,1,4,9,3],
    [1,9,7,8,3,4,5,6,2],
    [8,2,6,1,9,5,3,4,7],
    [3,7,4,6,8,2,9,1,5],
    [9,5,1,7,4,3,6,2,8],
    [5,1,9,3,2,6,8,7,4],
    [2,4,8,9,5,7,1,3,6],
    [7,6,3,4,1,8,2,5,9],
]
print(is_sudoku(A))
True

B = [
    [4,3,5,2,6,9,7,8,1],
    [6,8,2,5,7,1,4,9,3],
    [1,9,7,8,3,4,5,6,2],
    [8,2,6,1,9,5,3,4,7],
    [3,7,4,6,8,2,9,1,5],
    [9,5,1,7,4,3,6,2,8],
    [5,1,9,3,2,6,8,7,4],
    [2,4,8,9,5,7,1,3,6],
    [7,6,3,4,1,8,2,5,0],
]  # not a Sudoku; the last element is 0
print(is_sudoku(B))
False
```

```
C = [
    [5,3,4,6,7,8,9,1,2],
    [6,7,2,1,9,5,3,4,8],
    [1,9,8,3,4,2,5,6,7],
    [8,5,9,7,6,1,4,2,3],
    [4,2,6,8,5,3,7,9,1],
    [7,1,3,9,2,4,8,5,6],
    [9,6,1,5,3,7,2,8,4],
    [2,8,7,4,1,9,6,3,5],
    [3,4,5,2,8,6,1,7,9],
]
print(is_sudoku(C))
True
```