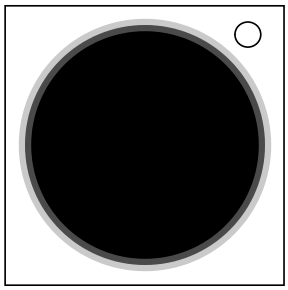
On the Subject of Jank Hole



Holeker.

This module contains a Jank Hole, which will flash a sequence of 10 colors which can be **Red**, **Green**, **Blue**, **Cyan**, **Magenta**, **Yellow** or **White**. There will be a pause between the start and the end of the sequence, where a swamp green color will be shown.

To solve the module, obtain a code consisting of letters A-Z.

In this module, when doing operations with letters, use their alphabetic positions, where Z = 0, A = 1, B = 2... Y = 25, same with converting numbers to letters.

First, using the serial number, find the goal letters X, Y and Z.

- If there are two letters in the serial number:
  - The first two letters of the serial number are X and Y, to find Z, take the absolute difference between X and Y, then divide it by 2, if there are more unlit indicators than lit indicators, round down, otherwise, round up. Then convert the result into a letter using alphabetic position, it will be letter Z.
- If there are three letters in the serial number:
  - The serial number letters are going to be X, Y and Z.
- If there are four letters in the serial number:
  - Find the absolute difference between the 1st and 2nd letter of serial number, 2nd and 3rd, 3rd and 4th. Convert the results into letters using alphabetic position, in order the letters are going to be X, Y and Z.

Then, figure out the two keys: **Key A** and **Key B**, using the color sequence and the table below. For **Key A**, use the first 5 colors of the sequence in order, for **Key B** use the last 5 colors in order. Both of the Keys start as empty strings and the table will add letters to them or modify the keys. First, figure out **Key A**, then figure out **Key B**, as **Key A** may be used in calculating **Key B**.

Color\Key	Key A	Key B
Red	If last digit of the serial number is even, append "EVEN" to **Key A**, otherwise append "ODD" to **Key A**.	Reverse **Key B**.
Green	Rearrange **Key A** so that letters with odd positions are listed first, while the letters with even positions are listed after.	Reverse the second half of **Key B** (the half will contain last [(Length of **Key B**) / 2, rounded up] letters of **Key B**).
Blue	Reverse the first half of **Key A** (the half will contain first [(Length of **Key A**) / 2, rounded down] letters of **Key A**).	Append **Key A** to **Key B**.
Cyan	Reverse **Key A**.	Rearrange **Key B** so that letters with even positions are listed first, while the letters with odd positions are listed after.
Magenta	Append the name of each lit indicator in alphabetical order to **Key A**.	If the length of **Key A** is odd, append the letter in the center of **Key A** to **Key B**, otherwise append the first letter of **Key A** to **Key B**.
Yellow	Append the Goal Letters in order of X, Y and Z to **Key A**.	Append the name of each unlit indicator in alphabetical order to **Key B**.
White	Append all the serial number letters in order left to right to **Key A**.	Append all the serial number digits in order left to right, converted to a letter using the digits as the alphabetic positions to **Key B**.

After figuring out **Key A** and **Key B**, for each of the keys:

- Append the alphabet to the key.
- If there are duplicate letters, only keep the first occurrences.

Then, go through both of the keys left to right at the same time. If the current letters from both of the keys are the same, add 1 to **Key B**'s letter at the current position, looping back to Z at 26. These are your final **Key A** and **Key B**.

Next, use your goal letters to create a binary grid, which will always have 3 rows.

Take your goal letters and split them into three rows, then translate each letter into morse code. Then, convert each morse into binary where:

- A dash is represented by "111".
- A dot is represented by "1".
- A space between each dot and dash character is represented by "0".

If there is a row that is shorter than the others, append 0s to that row until it reaches the length of the longest row.

Then, create a zeros grid and a ones grid in the paragraph below using the Keys you made, where **Key A** will be used to make the zeros grid and **Key B** will be used to make the ones grid.

Create an empty grid with the same size as the binary grid and fill it with the key in reading order, looping back to the start of the key when reaching the end.

The Jank Hole will submit letters in braille reading order (Going through each column top to bottom, left to right). Refer to "Appendix - BHSCII Table" for how to enter letters. In the table, a dot means a color switch in the sequence, a vertical line line means that you need to press and immediately release the hole, and square brackets mean holding the hole for one or more color switches. Your input will be automatically submitted in the pause between the start and the end of the sequence, and the letter that was inputted will be shown. If it's not a valid letter, a "?" will be shown. The [ppp] gesture (a hold for 3 color switches) will display the number of entered letters in the pause, accounting for the skipped letters (If you do the [ppp] gesture after entering one letter and solving one module, 4 will be displayed).

Submitting the letter at the current position in the zeros grid will submit a zero, and submitting the letter at the current position in the ones grid will submit a one, while submitting a letter which is from neither of the grids at the current position will make the module strike.

Submit letters to recreate the binary grid. Submitting a 1 while the number at the current position in the binary grid is a 0, and vice versa, will also make the module strike, otherwise the module will accept the input. The letters you need to submit for each position of the binary grid in braille reading order will form the **Solution Code**. The module will be solved once the correct letter is submitted for the **8th** position of the **Solution Code**.

If you solve a non-Jank Hole module on the bomb after entering the correct letter for the binary grid the amount of numbers that need to be entered will be shortened by 3 (so, next 3 digits of the binary grid will be skipped), unless the code is currently 3 or less characters long, in which case you will have to enter the last letter of the code.