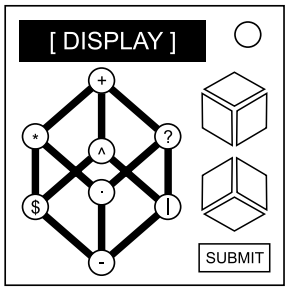
On the Subject of Regular Hexpressions



You can't parse [X]HTML with regex. Regex is not a tool that can be used to correctly parse HTML. Every time you attempt to parse HTML with regular expressions, the unholy child weeps the blood of virgins, and Russian hackers pwn your alarm clock. If you parse HTML with regex you are giving in to Them and their blasphemous ways which doom us all to inhuman toil for the One whose Name cannot be expressed in the Basic Multilingual Plane, he comes. A mere glimpse of the world of regex parsers for HTML will instantly transport a programmer's consciousness *into a world of ceaseless screaming, he comes, ~~the pestilent slithy~~ regex-infection will devour your HTML parser, application and existence for all time.*

Have you tried using regex to defuse a bomb instead?

- This module has eight Vertices representing the corners of a regular hexahedron, and six Face buttons corresponding to the sides of the hexahedron. Pressing a Face button will cycle the corresponding Vertices. Each Vertex is labeled with a symbol for your convenience; the labels play no role in solving the module.
- Each Vertex has an associated word, which is shown on the display when the Vertex is at the top of the hexahedron. Use the rules below and Table 1 to determine the four target regular expressions (target regexes), then find the Vertex words that match each target regex[1]. To solve the module, arrange the Vertex words so that the topmost word matches the first target regex, the second-topmost word matches the second target regex, and so on to the fourth-topmost (bottom) word; then press the Submit button. The arrangement of the two Vertex words on the left and the two on the right do not matter.
- Note that there may be multiple words matching a given target regex, but there is only one way to arrange the Vertex words so that each word going down the center of the hexahedron matches the corresponding target regex.

[1] For a word to match a regex, the entire word must match. Page 3 explains the full mechanics of regex matching. If you are familiar with regexes, you should consider every regex to have an implicit `^` at the beginning and an implicit `$` at the end.

Target Regular Expressions

First, determine the initial row and initial column, shown on the display when the top Vertex is moving, and labeled "R" and "C", respectively.

Next, compute the row offset as the number of solvable (non-needy) modules modulo 11, plus 1. Add this to the initial row modulo 12 to obtain the final row. Similarly, compute the column offset as the number of batteries and indicators modulo 5, plus 1. Add this to the initial row modulo 6 to obtain the final column.

Finally, use Table 1 to find the four regexes that are in either the initial or final row, and in either the initial or final column. The target regexes are ordered clockwise starting from the north-west: the first target regex is the north-west one, the second is the north-east, the third is the south-east, and the fourth is the south-west.

	0	1	2	3	4	5
0	`.*O.*H.*`	`....`	`.*[^O]N.*`	`A.*`	`.*O.*E.*`	`.*(ER|RE).*`
1	`.*R[^E].*`	`[^ASTERISK]+`	`.*[AEIOU].[AEIOU].*`	`.*(OW|WO).*`	`.*M[^O].*`	`.*W(HA|HI|A).*`
2	`.*O`	`.*OT.*`	`....?`	`.*[OU][MN].*`	`.*E`	`.*F.*T.*`
3	`.....`	`.*[^O]T.*`	`D.*`	`.*I.*T.*`	`.*(FI|IF).*`	`.*F[^O].*`
4	`[^BRACKET]+`	`.*[^AEIOU][^AEIOU].*`	`.*(NO|ON).*`	`.*U[^G].*`	`.*OL?[DT].*`	`.*[AEIOU][AEIOU].*`
5	`.*HT.*`	`.....?`	`.*[BT][OE].*`	`.*H`	`.*OW.*`	`......`
6	`.*[^H]T.*`	`H.*`	`.*B.*T.*`	`..`	`.*B[^O].*`	`[^REFUTE]+`
7	`.*[^AEIOU][^AEIOU].`	`.*(OT|TO).*`	`.*O[^W].*`	`[^STRAIGHT]+`	`.*[AEIOU].*[AEIOU].*`	`.*TH.*`
8	`(..)?.`	`.*L[AEIOU].*`	`.*P`	`.*ON.*`	`..?`	`.*[^E]A.*`
9	`W.*`	`.*F.*R.*`	`...`	`.*I[^N].*`	`[^ELIMINATE]+`	`[^AEIOU][^AEIOU].*`
10	`.*(HT|TH).*`	`.*E[^A].*`	`[^QUESTION]+`	`.*[AEIOU]`	`.*DI.*`	`.*O[^T].*`
11	`.[AEIOU].`	`.*Y`	`.*OR.*`	`...?`	`.*[^S]T.*`	`U.*`

Table 1: Regular expressions.
Some regexes are split across multiple lines in their cells.
For example, the regex in row 4, column 1 is `.*[^AEIOU][^AEIOU].*`.

Matching Regular Expressions

A word matches a regular expression if each part of the regex corresponds to a matching part of the word, in order and without gaps. Each regex part consists of a Basic Element and an optional Quantifier.

Basic Elements

- An individual letter in a regex matches an individual letter in the word. Examples: `A` matches the A part in MATCH, and `E` matches each E part in SEE.
- A period matches any individual character in the word. Example: `.` matches the M, A, T, C, and H parts in MATCH.
- Letters in square brackets match any one of the letters. Example: `[CAT]` matches the A, T, and C parts in MATCH.
- A caret and letters in square brackets match any character except those letters. Example: `[^CAT]` matches the M and H parts in MATCH.
- A pipe-separated list of strings surrounded by parentheses matches any individual string in the list. Example: `(H|AT)` matches the AT and H parts in MATCH.

Quantifiers

A Basic Element without a Quantifier matches exactly one part of a word. With a Quantifier, the Basic Element can match zero or multiple consecutive parts of a word, depending on the quantifier.

- The question-mark quantifier matches zero or one of the Basic Element. Example: `CH?` matches the initial C and the CH part in CATCH.
- The asterisk quantifier matches zero or more of the Basic Element. Example: `A[CTM]*` matches the A, AT, and ATC parts in MATCH.
- The plus quantifier matches one or more of the Basic Element. Example: `A[CTM]+` matches the AT and ATC parts in in MATCH.

More Examples

- `A[ES]+` matches ASSESS. It does not match SEE because `A` has no matching part of the word, and it does not match A because `SEE` has no matching word part.
- `(SUB|CON)TRACTS?` matches SUBTRACT, CONTRACT, SUBTRACTS, and CONTRACTS. It does not match TRACT because `(SUB|CON)` has no matching word part.
- `I.*[^AK]` matches any word starting with an I and ending with any letter except A or K, such as IT or IMPLY or ICONIC, but not INK or IDEA. It does not match the word I because the `[^AK]` has no matching word part.