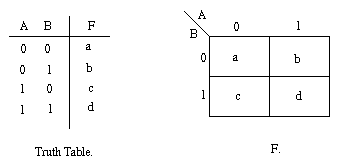
**K-Map Solver:**

**Explanation**:

A Karnaugh map (K-map) is a pictorial method used to minimize [Boolean](https://whatis.techtarget.com/definition/Boolean) expressions without having to use Boolean algebra theorems and equation manipulations. A K-map can be thought of as a special version of a [truth table](https://whatis.techtarget.com/definition/truth-table).



Our aim with this project is to simplify and automate the process of solving a K-map for significantly less time and accurate results.

* Steps to solve a K-map: *Convention method*

1. Select K-map according to the number of variables.
2. Identify minterms or maxterms as given in the problem.
3. For SOP put 1’s in blocks of K-map respective to the minterms (0’s elsewhere).
4. For POS put 0’s in blocks of K-map respective to the maxterms(1’s elsewhere).
5. Make rectangular groups containing total terms in the power of two like 2,4,8 ..(except 1) and try to cover as many elements as you can in one group.
6. From the groups made in step 5 find the product terms and sum them up for the SOP form

**Algorithm**:

* **Input:**
  1. We take Input for K-map type, number of variables, number of 1s, and “don't care” in a 4x4 matrix (ar[4][4])
* **Grouping:**
  1. Checking for a group of 16, 8, 4, 2, and single 1s (or “don’t care”s)
     1. We iterate over all the squares and check if a group of 16, 8, 4, 2, and single 1s can be formed.
     2. We save that information into a stack and proceed.
  2. Case 1: groups of 16 1s
     1. We will use a loop to iterate over all the indices in the K-map array.
     2. If all the indices have 1s then it is grouped as a group of 16 1s.
  3. Case 2: groups of 8 and 4 1s
     1. We will use a loop to iterate over all the positions in the kmap and check if each position satisfies the following conditions where i,j are the respective matrix indices for the K-map:
        + If ar[ i+1 ][ j ] == 1
        + If ar[ i ][ j+1] == 1
        + If ar[ i+1 ][ j ] and ar[ i ][ j+1 ] == 1
     2. If all the three conditions are satisfied, then the respective groups of 8 or 4 are formed and saved into the stack.
  4. Case 3: for groups of 4 and 2 1s
     1. We will use a loop to iterate over all the positions in the k-map and check if each position satisfies the following conditions where i,j are the respective matrix indices for the K-map:
        + If ar[ i+1 ][ j ] == 1, or
        + If ar[ i ][ j+1] == 1
     2. If all the three conditions are satisfied, then the respective groups of 4 and 2 are formed and saved into the stack.
  5. Case 4: for groups of single 1s
     1. All the 1s that are left in the k-map will be stored as a group of single 1s.

**Note:** While checking for the given conditions, the indices may go out of bound therefore we have to correct the index value each time we check the conditions. To do that, instead of using “i” and “j” directly, we can use “4%i” and “4%j”.

For example, if “i” becomes 4 it will be converted to 4%4 = 0, therefore i = 0

* **Selection:** 
  1. We eliminate all the smaller groups which are a subset of any other larger group.
  2. We eliminate the groups of two 1s which don't contain at least a single 1 that is not in any other group.
* **Output:** 
  1. With all the selected groups we’ll generate the corresponding boolean expression and print the output.

**Input**:

1. Type of K-map ([SOP](https://www.electronics-tutorials.ws/boolean/sum-of-product.html) or [POS](https://www.electronics-tutorials.ws/boolean/product-of-sum.html))
2. Number of variables for the K-map (2 variable, 3 variable, etc)
3. Position of min-terms or max-terms.
4. Position of “Don’t Care”s (if any)

**Output**:

1. A minimized Boolean expression. (eg: AB’ + C)