1. There is a 2D array, answer some questions

```
int array_2d[2][3] = {1, 2, 3, 4, 5, 6};
```

   a. What's the meaning of &array_2d, array_2d, &array_2d[0], array_2d[0], &array_2d[0][0]?

Expression	Type	Meaning	Value
&array_2d	int (*)[2][3]	Address of the whole 2D array	Address of array_2d
array_2d	int (*)[3]	Address of the first row	Address of array_2d[0]
&array_2d[0]	int (*)[3]	Address of the first row	Address of array_2d[0]
array_2d[0]	int *	Pointer to first element of row 0	Address of array_2d[0][0]
&array_2d[0][0]	int *	Address of first element	Address of array_2d[0][0]

   b. What is the relationship between a 2D array and a pointer to array?

   A 2D array is a real contiguous block of memory. A pointer to an array (int (*)[3]) is a pointer storing the address of a row. Both of them access by subscript, (arra_2d[i][j], prt[i][j]). An array name decays to a pointer to its first row: int (*)[3] and it is not modifiable. Pointer to array stores the address of a row and it is modifiable.

   c. How to access each element of the two-dimensional array through a pointer to array?

```c
#include <stdio.h>

int main() {
    int array_2d[2][3] = { {1, 2, 3}, {4, 5, 6} };
    int (*ptr)[3] = array_2d;              // Pointer to an array of 3 integers


    for (int i = 0; i < 2; i++) {          // Access each element using pointer arithmetic
        for (int j = 0; j < 3; j++) {
            printf("%d ", ptr[i][j]);      // Equivalent to array_2d[i][j]
        }
        printf("\n");
    }

    return 0;
}

```

2. Why do we need a pointer to array?
    a.
        1. To passes array address instead of copying entire array
        2. To reduces memory overhead when working with large arrays
        3. To allows modifying original array within functions

    What is it used for?

    b.

```c
// Clean Multi-dimensional Array Processing
int matrix[3][4];
int (*ptr)[4] = matrix;


// Allocate 2D array dynamically
int rows = 3, cols = 4;
int (*dynamic)[4] = malloc(rows * sizeof(*dynamic));
```

```c
3
4   // argument of a function
5   void processMatrix(int (*arr)[4], int rows) {
6       // Can use array notation
7       for(int i = 0; i < rows; i++) {
8           for(int j = 0; j < 4; j++) {
9               arr[i][j] *= 2;  // Direct array access
10          }
11      }
12  }
13
```

3. Pointers-On-C.pdf 8.7 11

Expression	Value	Type of x
array	1000	int (*x)[2][3][6]
array + 2	1120	int (*x)[2][3][6]
array[3]	11b0	int (*x)[3][6]
array[2] - 1	10d8	int (*x)[3][6]
array[2][1]	1168	int (*x)[6]
array[1][0] + 1	10a8	int (*x)[6]
array[1][0][2]	10c0	int *x
array[0][1][0]	1050	int *x
array[3][1][2][5]	garbage	int x
&array[3][1][2][5]	123c	int *x

4. Pointers-On-C.pdf 8.7 13
    a.

Pointer Expression	Array Subscript
*array	array[0]
*(array + 2)	array[2]
*(array + 1) + 4	&array[1][4]
((array + 1) + 4)	array[1][4]
((*(array + 3) + 1) + 2)	array[3][1][2]
((array + 1) + 2)	array[1][2]
*(**array + 2)	array[0][0][2]
**(*array + 1)	array[0][1][0]
***array	array[0][0][0]