1.

Create a program in the following way:

- In the `main` function, declare a static two-dimensional array **B** with N rows and M columns. Use the directive `#define` of a preprocessor.
- Write a function which calculates values of the elements of the array according to the formula (i – row index, j – column index): $B_{ij} = (i+1)*(j+1)$. Create a header of the function in the following way:

```
void init_tab(int N, int M, double B[][M]);
```

- Write a function which prints the array to the screen.
- Write a function which calculates an average of all elements of the array. The result should be returned by the function and printed inside the `main` function.
- Write a function which calculates numbers of the elements that are greater than the average and less than the average. Use pointers to store the obtained numbers in the variables declared inside the `main` function. Print the obtained values inside the `main` function.
- Write a function which stores the n-th row of the array **B** inside a static one-dimensional array declared inside the `main` function. Use a header:

```
void select_row(int N, int M, double B[][M], double row[]);
```

- Write a function which prints the n-th row obtained by using the function from the previous point.

2.

Modify the program in order to use a dynamic allocation to store the arrays:

- one-dimensional

```
// allocate a block of memory for elements
double *row = (double *) malloc(M * sizeof(double));

// free memory
free(row);
```

- two-dimensional

```
// allocate a continuous block of memory for all elements
double *elems = (double *) malloc(N * M * sizeof(double));
// allocate memory for the array storing pointers to the rows
double **B = (double **) malloc (N * sizeof(double));
// assign proper addresses of the rows to the pointers
for (int i = 0; i < N; i++)
{
  B[i] = &elems[i*M];
}
                                    // write a code here which uses the array
// free memory
free(B);                            // free memory used by the pointer array
free(elems);                        // free memory used by the elements
```