1) Write a c program to increase or decrease the existing size of an 1D array?

```
#include <stdio.h>

#include <stdlib.h>

int main()

{

    int *array, size, new_size, i;

    printf("Enter the size of the array: ");

    scanf("%d", &size);

    array = (int *)malloc(size * sizeof(int));

    printf("Enter the elements of the array: ");.

    for (i = 0; i < size; i++)

    {

        scanf("%d", &array[i]);

    }

   printf("Enter the new size of the array: ");

    scanf("%d", &new_size);

    if (new_size > size)

    {

        array = (int *)realloc(array, new_size * sizeof(int));

        for (i = size; i < new_size; i++)

        {
            array[i] = 0;
        }
    }
    else if (new_size < size)
    {
        array = (int *)realloc(array, new_size * sizeof(int));
    }

    printf("The new array is: ");
    for (i = 0; i < new_size; i++)
    {
        printf("%d ", array[i]);
    }
    free(array);
    return 0;
}
```

Explanation : This C program prompts the user to input the size of an array, then takes the elements of the array as input. After that, it asks for a new size for the array and reallocates memory accordingly. If the new size is greater than the original size, the additional elements are initialized to 0. If the new size is less than the original size, the array is simply reallocated without initializing the new elements. Finally, it prints the elements of the modified array and frees the allocated memory.

Here's a brief breakdown of the program:

1. **array**: Pointer to the dynamically allocated array.
2. **size**: Variable to store the original size of the array.
3. **new_size**: Variable to store the new size of the array.
4. **i**: Loop variable.

The program uses dynamic memory allocation functions (**malloc** and **realloc**) to create and resize the array. It also uses **free** to release the allocated memory when done.

A few points to note:

- The program initializes additional elements to 0 only when the new size is greater than the original size.
- When decreasing the size of the array, the elements beyond the new size are not explicitly set to 0. This may result in unpredictable values in those locations.

Overall, the program is a simple example of dynamic memory allocation and reallocation in C.

2) Write a c program on 2D array to Increase & Decrease
i) No of subarrays

```
#include <stdio.h>

#include <stdlib.h>

int main() {

   int **matrix, rows, cols, i, j;

   // Initialize the 2D array
```

```
printf("Enter the number of rows: ");
scanf("%d", &rows);
printf("Enter the number of columns: ");
scanf("%d", &cols);
matrix = (int **)malloc(rows * sizeof(int *));
for (i = 0; i < rows; i++) {
    matrix[i] = (int *)malloc(cols * sizeof(int));
}
// Initialize the elements of the matrix
printf("Enter the elements of the matrix:\n");
for (i = 0; i < rows; i++) {
    for (j = 0; j < cols; j++) {
        scanf("%d", &matrix[i][j]);
    }
}
// Display the original matrix
printf("Original Matrix:\n");
for (i = 0; i < rows; i++) {
    for (j = 0; j < cols; j++) {
 ,       printf("%d\t", matrix[i][j]);
    }
    printf("\n");
    }
    // Increase or decrease the number of subarrays
    int choice;
    printf("\nEnter 1 to increase the number of subarrays, 2 to decrease: 
");
    scanf("%d", &choice);
    if (choice == 1) {
        // Increase the number of subarrays
        int newRows;
        printf("Enter the number of additional rows: ");
        scanf("%d", &newRows);
        matrix = (int **)realloc(matrix, (rows + newRows) * sizeof(int *));
        for (i = rows; i < rows + newRows; i++) {
            matrix[i] = (int *)malloc(cols * sizeof(int));
        }
        rows += newRows;
    } else if (choice == 2) {
        // Decrease the number of subarrays
        int removeRows;
        printf("Enter the number of rows to remove: ");
        scanf("%d", &removeRows);
        if (removeRows <= rows) {
          matrix = (int **)realloc(matrix, (rows - removeRows) * sizeof(int
*));
          rows -= removeRows;
       } else {
          printf("Error: Cannot remove more rows than present.\n");
       }
    } else {
       printf("Invalid choice.\n");
    }
    // Display the modified matrix
    printf("\nModified Matrix:\n");
    for (i = 0; i < rows; i++) {
       for (j = 0; j < cols; j++) {
          printf("%d\t", matrix[i][j]);
       }
       printf("\n");
    }
    // Deallocate memory
   for (i = 0; i < rows; i++) {
       free(
matrix[i]);-
    }
    free(matrix);
    return 0;
}
```

ii) elements in the subarrays

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int **matrix, rows, cols, i, j;
    // Initialize the 2D array
    printf("Enter the number of rows: ");
    scanf("%d", &rows);
    printf("Enter the number of columns: ");
    scanf("%d", &cols);
    matrix = (int **)malloc(rows * sizeof(int *));
    for (i = 0; i < rows; i++) {
        matrix[i] = (int *)malloc(cols * sizeof(int));
    }
    // Initialize the elements of the matrix
    printf("Enter the elements of the matrix:\n");
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
    // Display the original matrix
    printf("Original Matrix:\n");
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
    // Increase or decrease the number of elements in subarrays
    int choice;
    printf("\nEnter 1 to increase the number of elements, 2 to decrease:
");
    scanf("%d", &choice);
    if (choice == 1) {
        // Increase the number of elements
        int newCols;
        printf("Enter the number of additional columns: ");
        scanf("%d", &newCols);
        for (i = 0; i < rows; i++) {
            matrix[i] = (int *)realloc(matrix[i], (cols + newCols) * sizeof(int));
        }
        cols += newCols;
    } else if (choice == 2) {
        // Decrease the number of elements
        int removeCols;
        printf("Enter the number of columns to remove: ");
        scanf("%d", &removeCols);
        if (removeCols <= cols) {
            for (i = 0; i < rows; i++) {
                matrix[i] = (int *)realloc(matrix[i], (cols - removeCols) *
sizeof(int));
            }
            cols -= removeCols;
        } else {
            printf("Error: Cannot remove more columns than present.\n");
        }
    } else {
        printf("Invalid choice.\n");
    }
    // Display the modified matrix
    printf("\nModified Matrix:\n");
    for (i = 0; i < rows; i++) {
        for (j = 0; j < cols; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
    // Deallocate memory
    for (i = 0; i < rows; i++) {
        free(matrix[i]);
    }
    free(matrix);
    return 0;
}
```