```c
#include <stdio.h>
#include <stdlib.h>

#define NUM_ROWS 5
#define NUM_COLUMNS 3

int main(void)
{
    //variable declarations
    int i, j;
    int **ptr_iArray = NULL;

    //code
    // *** EVERY ROW OF A 2D ARRAY IS AN INTEGER ARRAY ITSELF COMPRISING OF 'NUM_COLUMNS' INTEGER ELEMENTS ***
    // *** THERE ARE 5 ROWS AND 3 COLUMNS IN A 2D INTEGER ARRAY. EACH OF THE 5 ROWS IS A 1D ARRAY OF 3 INTEGERS.
    // *** HENCE, EACH OF THESE 5 ROWS THEMSELVES BEING ARRAYS, WILL BE THE BASE ADDRESSES OF THEIR RESPECTIVE ROWS ***
    printf("\n\n");

    // *** MEMORY ALLOCATION ***
    ptr_iArray = (int **)malloc(NUM_ROWS * sizeof(int *)); //ptr_iArray is the name and base address of 1D Array containing 5 integer pointers to 5 integer arrays ... so it is an array containing elelments of data type (int *)
    if (ptr_iArray == NULL)
    {
        printf("MEMORY ALLOCATION TO THE 1D ARRAY OF BASE ADDRESSES OF %d ROWS FAILED !!! EXITTING NOW...\n\n", NUM_ROWS);
        exit(0);
    }
    else
        printf("MEMORY ALLOCATION TO THE 1D ARRAY OF BASE ADDRESSES OF %d ROWS HAS SUCCEEDED !!!\n\n", NUM_ROWS);

    // *** ALLOCATING MEMORY TO EACH ROW ***
    for (i = 0; i < NUM_ROWS; i++)
    {
        ptr_iArray[i] = (int *)malloc(NUM_COLUMNS * sizeof(int)); //ptr_iArray [i] is the base address of ith row ...
        if (ptr_iArray == NULL)
        {
            printf("MEMORY ALLOCATION TO THE COLUMNS OF ROW %d FAILED !!! EXITTING NOW...\n\n", i);
            exit(0);
        }
        else
            printf("MEMORY ALLOCATION TO THE COLUMNS OF ROW %d HAS SUCCEEDED !!!\n\n", i);
    }

    // *** ASSIGNING VALUES ***
    for (i = 0; i < NUM_ROWS; i++)
    {
        for (j = 0; j < NUM_COLUMNS; j++)
```

```c
        {
            *(*(ptr_iArray + i) + j) = (i + 1) * (j + 1); // ptr_iArray[i][j] = (i + 1) * (j + 1);
        }
    }

    // *** DISPLAYING VALUES ***
    printf("\n\n");
    printf("2D Integer Array Elements Along With Addresses : \n\n");
    for (i = 0; i < NUM_ROWS; i++)
    {
        for (j = 0; j < NUM_COLUMNS; j++)
        {
            printf("ptr_iArray_Row[%d][%d] = %d \t \t At Address &ptr_iArray_Row[%d][%d] : %p\n", i, j, ptr_iArray[i][j], i, j, &ptr_iArray[i][j]);
        }
        printf("\n\n");
    }

    // *** FREEING ALLOCATED MEMORY ***
    // *** FREEING MEMORY OF EACH ROW ***
    for (i = (NUM_ROWS - 1); i >= 0; i--)
    {
        if (*(ptr_iArray + i)) // if(ptr_iArray[i])
        {
            free(*(ptr_iArray + i)); // free(ptr_iArray[i])
            *(ptr_iArray + i) = NULL; // ptr_iArray[i] = NULL;
            printf("MEMORY ALLOCATED TO ROW %d HAS BEEN SUCCESSFULLY FREED !!! \n\n", i);
        }
    }

    // *** FREEING MEMORY OF ptr_iArray WHICH IS THE ARRAY OF 5 INTEGER POINTERS ... THAT IT, IT IS AN ARRAY HAVING 5 INTEGER ADDRESSES (TYPE int *) ***
    if (ptr_iArray)
    {
        free(ptr_iArray);
        ptr_iArray = NULL;
        printf("MEMORY ALLOCATED TO ptr_iArray HAS BEEN SUCCESSFULLY FREED !!! \n\n");
    }

    return(0);
}
```