

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define NUM_ROWS 5
5  #define NUM_COLUMNS 3
6
7  int main(void)
8  {
9      //variable declarations
10     int i, j;
11     int **ptr_iArray = NULL;
12
13     //code
14     // *** EVERY ROW OF A 2D ARRAY IS AN INTEGER ARRAY ITSELF COMPRISING OF
15     // 'NUM_COLUMNS' INTEGER ELEMENTS ***
16     // *** THERE ARE 5 ROWS AND 3 COLUMNS IN A 2D INTEGER ARRAY. EACH OF THE 5
17     // ROWS IS A 1D ARRAY OF 3 INTEGERS.
18     // *** HENCE, EACH OF THESE 5 ROWS THEMSELVES BEING ARRAYS, WILL BE THE
19     // BASE ADDRESSES OF THEIR RESPECTIVE ROWS ***
20     printf("\n\n");
21
22     // *** MEMORY ALLOCATION ***
23     ptr_iArray = (int **)malloc(NUM_ROWS * sizeof(int *)); //ptr_iArray is the
24     // name and base address of 1D Array containing 5 integer pointers to 5
25     // integer arrays ... so it is an array containing elements of data type
26     // (int *)
27     if (ptr_iArray == NULL)
28     {
29         printf("MEMORY ALLOCATION TO THE 1D ARRAY OF BASE ADDRESSES OF %d ROWS
30         FAILED !!! EXITTING NOW...\n\n", NUM_ROWS);
31         exit(0);
32     }
33     else
34     {
35         printf("MEMORY ALLOCATION TO THE 1D ARRAY OF BASE ADDRESSES OF %d ROWS
36         HAS SUCCEEDED !!!\n\n", NUM_ROWS);
37     }
38
39     // *** ALLOCATING MEMORY TO EACH ROW ***
40     for (i = 0; i < NUM_ROWS; i++)
41     {
42         ptr_iArray[i] = (int *)malloc(NUM_COLUMNS * sizeof(int)); //ptr_iArray
43         // [i] is the base address of ith row ...
44         if (ptr_iArray[i] == NULL)
45         {
46             printf("MEMORY ALLOCATION TO THE COLUMNS OF ROW %d FAILED !!!
47             EXITTING NOW...\n\n", i);
48             exit(0);
49         }
50         else
51         {
52             printf("MEMORY ALLOCATION TO THE COLUMNS OF ROW %d HAS
53             SUCCEEDED !!!\n\n", i);
54         }
55     }
56
57     // *** ASSIGNING VALUES ***
58     for (i = 0; i < NUM_ROWS; i++)
59     {
60         for (j = 0; j < NUM_COLUMNS; j++)
```

```
46     {
47         *(*(ptr_iArray + i) + j) = (i + 1) * (j + 1); // ptr_iArray[i][j] = ?
            (i + 1) * (j + 1);
48     }
49 }
50
51 // *** DISPLAYING VALUES ***
52 printf("\n\n");
53 printf("2D Integer Array Elements Along With Addresses : \n\n");
54 for (i = 0; i < NUM_ROWS; i++)
55 {
56     for (j = 0; j < NUM_COLUMNS; j++)
57     {
58         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]);
59     }
60     printf("\n\n");
61 }
62
63 // *** FREEING ALLOCATED MEMORY ***
64 // *** FREEING MEMORY OF EACH ROW ***
65 for (i = (NUM_ROWS - 1); i >= 0; i--)
66 {
67     if (*(ptr_iArray + i)) // if(ptr_iArray[i])
68     {
69         free(*(ptr_iArray + i)); // free(ptr_iArray[i])
70         *(ptr_iArray + i) = NULL; // ptr_iArray[i] = NULL;
71         printf("MEMORY ALLOCATED TO ROW %d HAS BEEN SUCCESSFULLY FREED !!! ?
            \n\n", i);
72     }
73 }
74
75 // *** 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 ?
    *) ***
76 if (ptr_iArray)
77 {
78     free(ptr_iArray);
79     ptr_iArray = NULL;
80     printf("MEMORY ALLOCATED TO ptr_iArray HAS BEEN SUCCESSFULLY FREED !!! ?
        \n\n");
81 }
82
83 return(0);
84 }
85
86
87
88
89
```