

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define NUM_ROWS 5
5  #define NUM_COLUMNS_ONE 3
6  #define NUM_COLUMNS_TWO 8
7
8  int main(void)
9  {
10     //variable declarations
11     int *iArray[NUM_ROWS]; //A 2D Array which will have 5 rows and number of
12     //columns can be decided later on ...
13     int i, j;
14     //code
15
16     // ***** ONE (ALLOCATING MEMORY FOR AN ARRAY OF 3 INTEGERS PER ROW) *****
17     printf("\n\n");
18     printf("***** FIRST MEMORY ALLOCATION TO 2D INTEGER ARRAY *****\n\n");
19     for (i = 0; i < NUM_ROWS; i++)
20     {
21         iArray[i] = (int *)malloc(NUM_COLUMNS_ONE * sizeof(int));
22         if (iArray[i] == NULL)
23         {
24             printf("FAILED TO ALLOCATE MEMORY TO ROW %d OF 2D INTEGER\n\n", i);
25             exit(0);
26         }
27         else
28             printf("MEMORY ALLOCATION TO ROW %d OF 2D INTEGER ARRAY\n\n", i);
29     }
30
31     //ASSIGNING VALUES TO 2D ARRAY ...
32     for (i = 0; i < NUM_ROWS; i++)
33     {
34         for (j = 0; j < NUM_COLUMNS_ONE; j++)
35         {
36             iArray[i][j] = (i + 1) * (j + 1);
37         }
38     }
39
40     //DISPLAYING 2D ARRAY ...
41     printf("\n\n");
42     printf("DISPLAYING 2D ARRAY : \n\n");
43     for (i = 0; i < NUM_ROWS; i++)
44     {
45         for (j = 0; j < NUM_COLUMNS_ONE; j++)
46         {
47             printf("iArray[%d][%d] = %d\n", i, j, iArray[i][j]);
48         }
49         printf("\n\n");
50     }
51     printf("\n\n");
```

```
52
53 //FREEING MEMORY ASSIGNED TO 2D ARRAY (MUST BE DONE IN REVERSE ORDER)
54 for (i = (NUM_ROWS - 1); i >= 0; i--)
55 {
56     free(iArray[i]);
57     iArray[i] = NULL;
58     printf("MEMORY ALLOCATED TO ROW %d OF 2D INTEGER ARRAY HAS BEEN  \n
59     SUCCESSFULLY FREED !!!\n\n", i);
60 }
61 // ***** TWO (ALLOCATING MEMORY FOR AN ARRAY OF 8 INTEGERS PER ROW) *****
62 printf("\n\n");
63 printf("***** SECOND MEMORY ALLOCATION TO 2D INTEGER ARRAY ***** \n\n");
64 for (i = 0; i < NUM_ROWS; i++)
65 {
66     iArray[i] = (int *)malloc(NUM_COLUMNS_TWO * sizeof(int));
67     if (iArray[i] == NULL)
68     {
69         printf("FAILED TO ALLOCATE MEMORY TO ROW %d OF 2D INTEGER  \n
70         ARRAY !!! EXITTING NOW...\n\n", i);
71         exit(0);
72     }
73     else
74         printf("MEMORY ALLOCATION TO ROW %d OF 2D INTEGER ARRAY  \n
75         SUCCEEDED !!!\n\n", i);
76 }
77 //ASSIGNING VALUES TO 2D ARRAY ...
78 for (i = 0; i < NUM_ROWS; i++)
79 {
80     for (j = 0; j < NUM_COLUMNS_TWO; j++)
81     {
82         iArray[i][j] = (i + 1) * (j + 1);
83     }
84 }
85 //DISPLAYING 2D ARRAY ...
86 printf("\n\n");
87 printf("DISPLAYING 2D ARRAY : \n\n");
88 for (i = 0; i < NUM_ROWS; i++)
89 {
90     for (j = 0; j < NUM_COLUMNS_TWO; j++)
91     {
92         printf("iArray[%d][%d] = %d\n", i, j, iArray[i][j]);
93     }
94     printf("\n\n");
95 }
96 printf("\n\n");
97
98 //FREEING MEMORY ASSIGNED TO 2D ARRAY (MUST BE DONE IN REVERSE ORDER)
99 for (i = (NUM_ROWS - 1); i >= 0; i--)
100 {
101     free(iArray[i]);
102     iArray[i] = NULL;
```

```
103     printf("MEMORY ALLOCATED TO ROW %d OF 2D INTEGER ARRAY HAS BEEN  
        SUCCESSFULLY FREED !!!\n\n", i);
```

```
104     }
```

```
105
```

```
106     return(0);
```

```
107 }
```

```
108
```

```
109
```

```
110
```