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

#define NUM_ROWS 5
#define NUM_COLUMNS 5

int main(void)
{
    //variable declarations
    int *iArray[NUM_ROWS]; //A 2D Array which will have 5 rows and number of columns can be decided later on ...
    int i, j;

    //code
    printf("\n\n");
    printf("********** MEMORY ALLOCATION TO 2D INTEGER ARRAY **********\n\n");
    for (i = 0; i < NUM_ROWS; i++)
    {
        //ROW 0 WILL HAVE (NUM_COLUMNS - 0) = (5 - 0) = 5 COLUMNS...
        //ROW 1 WILL HAVE (NUM_COLUMNS - 1) = (5 - 1) = 4 COLUMNS...
        //ROW 2 WILL HAVE (NUM_COLUMNS - 2) = (5 - 2) = 3 COLUMNS...
        //ROW 3 WILL HAVE (NUM_COLUMNS - 3) = (5 - 3) = 2 COLUMNS...
        //ROW 4 WILL HAVE (NUM_COLUMNS - 4) = (5 - 4) = 1 COLUMN...

        //BEACUSE OF THIS, THERE IS NO CONTIGUOUS MEMORY ALLOCATION ... HENCE, ALTHOUGH WE MAY USE THE DATA AS A 2D ARRAY, IT IS NOT REALLY A 2D ARRAY IN MEMORY ...

        iArray[i] = (int *)malloc((NUM_COLUMNS - i) * sizeof(int));
        if (iArray[i] == NULL)
        {
            printf("FAILED TO ALLOCATE MEMORY TO ROW %d OF 2D INTEGER ARRAY !!! EXITTING NOW...\n\n", i);
            exit(0);
        }
        else
            printf("MEMORY ALLOCATION TO ROW %d OF 2D INTEGER ARRAY SUCCEEDED !!!\n\n", i);
    }

    for (i = 0; i < 5; i++)
    {
        for (j = 0; j < (NUM_COLUMNS - i); j++)
        {
            iArray[i][j] = (i * 1) + (j * 1);
        }
    }

    for (i = 0; i < 5; i++)
    {
        for (j = 0; j < (NUM_COLUMNS - i); j++)
        {
            printf("iArray[%d][%d] = %d \t At Address : %p\n", i, j, iArray[i][j], &iArray[i][j]);
        }
        printf("\n");
```

```
51      }
52
53      for (i = (NUM_ROWS - 1); i >= 0; i--)
54      {
55          if (iArray[i])
56          {
57              free(iArray[i]);
58              iArray[i] = NULL;
59              printf("MEMORY ALLOCATED TO ROW %d HAS BEEN SUCCESSFULLY FREED !!! ↵
                  \n\n", i);
60          }
61      }
62
63      return(0);
64  }
65
66
67
```