Ex: ~~int~~ *x;
x = calloc(n, sizeof(int)); could be used to define a 1-D array of integers of capacity n and x[0:n-1]=0

```
#define CALLOC(p,n,s)\
    if(!((p)=calloc(n,s))){ \
        fprintf(stderr, "Insufficient memory");
        exit(EXIT_FAILURE); \
    }
```

<u>Exercise 2.2</u>

1) Make the fewest no. of $\boxed{\text{changes}}$ to the function make2dArray so that it creates a two-dimensional array all of whose elements are set to 0.

?? not clear

```
int** make2DArray(int rows, int cols)
{
  int **x, i;
  MALLOC(x, rows*sizeof(*x));
  for(i=0; i<rows; i++)
    CALLOC(x[i], cols, sizeof(**x));
  return x;
}
```

I have only changed the MALLOC inside the for loop to CALLOC. The 1st parameter remains unchanged. But for CALLOC, there are 2 new params.
∴ Total of 4 changes.
But, this is optimal. ??

3) 1. void add(int a[][MAX_SIZE], int b[][MAX_SIZE], int c[][MAX_SIZE], int rows, int cols)

```
2. {
3.   int i,j;
4.   for(i=0; i<rows; i++)
5.     for(j=0; j<cols; j++)
6.       c[i][j] = a[i][j] + b[i][j];
7. }
```

<u>Inner Loop Invariant</u>

~~Before the iteration~~ Just before the start of the iteration when j=K, where 0≤K<cols, the elements of matrix c of the row i from index 0 to j-1 is the ~~correctly computed~~ corresponding sum of the matrix elements of a and b.

$\boxed{\text{Initialization:}}$ j=0. On line 5, j is initially 0. ∴ This is the beginning of 1st iteration. The indices 0 to -1 doesn't make sense. ∴ The loop invariant trivially holds.