Multi-dimensional Arrays

Following are some examples of declaring two dimension arrays:

```c
int A[5][8]; // 5 rows and 8 columns
char CH[10][20]; // 10 rows 20 columns
float X[2][2]; // 2 rows by 2 columns
```

Fill first example with 0’s

```c
for(int i=0; i<5; i++)
for(int j=0; j<8; j++)
A[i][j]=0;
```

Note that the outer index i represents the row numbers. So this routine is filling up the array with zeros, one row at a time. For each i, j runs from 0 to 7, which are the numbers for the columns.

Passing Multi-dimensional Arrays to Functions

Suppose we wish to set the previous example to zero’s by using a function call.

```c
Zero(A,5,8)
```

The header for this function will have a different representation for the array than for a one-dimensional array.

Function Definition

```c
void Zero(int A[][8], int r, int c){
Just as with the one-dimensional array, the first dimension is not included for A; however, the second one is.
Whenever a function is created that has a multi-dimensional array type as a parameter, all dimensions after the first must be included in the declaration within the parameter list.
In order to manipulate the array within the function, we also pass all lengths via simple integer parameter variables. In the case above these are r (rows) and c (columns).
```

Adding Columns

Suppose an array is declared by

```c
int X[10][10]
```

and all entries in the upper-left-hand 9X9 corner have been entered. Write code that will add up each row and each column placing the sums at the bottom of each column and right end of each row. Then write it as a function.