# CS33 Homework Assignment 2

*Due 11:59pm September 22, 2023*

1. Consider the following 2D array in C:

```
int A[M][N];
```

   a. We'd like to work with column 1 of the array, i.e., the data in A[0][1], A[1][1], A[2][1], etc. In particular, we want an *int* * that refers to a 1D array containing this column. Can this be done by setting such a pointer to point to the column's first element, or must we copy the elements of the column into a separate 1D array?

   b. We'd now like to work with row 1 of the array, i.e., the data in A[1][0], A[1][1], A[1][2], etc. In particular, we want an *int* * that refers to a 1D array containing this row. Can this be done by setting such a pointer to point to the row's first element, or must we copy the elements of the row into a separate 1D array?

2. We want a (3D) array of the 2D arrays of problem 1, i.e., we'd like to organize P MxN arrays as a single PxMxN array.
   a. How does one declare an array of P of the 2D arrays of problem 1?

   b. We would like a pointer *ptr* that refers to a 2D array (of problem 1), so that we can use it to iterate through the array of such 2D arrays. How would one declare such a pointer? (It's definitely not cheating to test your answer using gcc!)

   c. We would like a function *func* that takes an *int* as an argument and returns a pointer to our 2D array. How would one declare such a function?

3. What's wrong, if anything, with each of the following?
   a.

```
int proc(int m) {
     static int array[m];
     // ...
}
```

   b.

```
struct array_struct {
     int array[20];
};

struct array_struct init(void) {
     struct array_struct a_s;
     for (int i=0; i<20; i++)
          a_s.array[i] = i;
```

```
    return a_s;
}

int main(void) {
    struct array_struct x = init();
    // ...
}
```

c.

```
int main(int argc, char *argv[]) {
    int a=0, b=0;
    int c;

    if (argc != 3) {
        fprintf(stderr, "Wrong number of args\n");
        exit(1);
    }
    a = atoi(argv[1]);
    b = atoi(argv[2]);
    switch(a) {
    case 0:
        c=b;
        break;
    case 1:
        a=b;
        break;
    default:
        c=a;
    }
    return a+b+c;
}
```

d.

```
int *array;

void init(void) {
    int A[20];
    array = A;
}

int main(void) {
    init();
    array[7] = 6;
    // ...
}
```