1 Arrays

1. We create a two dimensional array with:

```c
double x[10][5]
```
or, better yet,

```c
double **x
```

2. Now we must allocate space to this `pointer to pointer to double`. We could start with

```c
x = (double**) malloc( (unsigned) (nrow * sizeof(double*)) );
```
which allocates space for a (column) vector of nrow pointers to double. Then we allocate space for each row:

```c
for(i=0;i<nrow;i++) x[i] = (double*) malloc((unsigned)(ncol*sizeof(double)));
```
or, better, we could first allocate all the memory in x[0] (assuring that the array will be stored contiguously). We do this as follows:

```c
xt = x[0] = (double *) malloc((unsigned)(nrow*ncol*sizeof(double)));
```
Then distribute the space across the other rows:

```c
for(i=1;i<nrow;i++) x[i]=(xt+=ncol);
```

Suppose we store a $5 \times 3$ matrix by columns. Then the (i,j)th element is in position $j*nrow+i$. That is, $x[i][j] = x[j*nrow+i]$. Instead of using so much subscripting, we should instead use pointer arithmatic.

Suppose we want the mean of each column of an $nrow \times ncol$ matrix, where we’ve declared:

```c
double *x;
x=(double *)malloc((unsigned)(nrow*ncol*sizeof(double)));
if(x==NULL) exit(1);
```
Could fill the matrix with:

```c
for(i=0;i<nrow;i++)
for(j=0;j<ncol;j++) scanf(‘%lf’,&x[j*nrow+i]);
```
Instead we should do it this way:
double *xt;
for (i=0; i<nrow; i++)
    xt = x + i;
for (j=0; j<ncol; j++, xt+=nrow)
    scanf("%lf", xt);

Now we compute the mean of the columns:

double *means;
double z;
means = (double *) malloc((unsigned)(ncol*sizeof(double)));
if (means == NULL) exit(1);
for (j=0; j<ncol; j++) {
    z = 0;
    for (i=0; i<nrow; i++)
        z += x[j*nrow + i];
    means[j] = z / (double)nrow;
}

But, better yet, we should do

double *xt;
xt = x;
for (j=0; j<ncol; j++)
    z = 0;
for (i=0; i<nrow; i++, xt++)
    z += *xt;
mean[j] = z / (double)nrow;