The core of the data is stored in a ragged 2-D array.

```
double **pointers //array of pointers of double type, each points too -->
pointers[x] //array of doubles

Functionally...
pointers[x] //pointer to an double array
pointers[x][y] //access a double in the array
```

```
double** readThisFile(double**);
```

Reads the file using a stringstream, dynamically allocates an double array to store files

Assign first integer to size of file to size of pointer array

Use stringstream to read subsequent lines

Dynamically allocate double arrays and read data into it

Add to pointer array

```
void printStuff(double**);
```

Prints out the table using a while loop and inner for loop

Outer while array to iterate array of pointers until NULL

Inner for loop for iterate array of doubles

Prints elements in table

```
void insertionSort(double[], int);
```

Sorts the individual double arrays in descending order using insertionSort

Changes:

1. *Modified to sort in descending order*
2. *Modified for the double type instead of int*
3. *Modified to ignore the first element of the array and start at the second*

```
double** tableSort(double**);
```

Sorts the array of pointers using an insertionSort, descending order

Changes:

1. *Modified to use two while loops instead of one for and one while*
2. *Modified to move pointers, but still compare doubles*
3. *Modified to sort in descending order*

```
int computeSize(double**);
```

Computes the total number of rows, aka number of rows in pointer array

While loop that counts row in 1-D array until it hits NULL, returns integer

```
void deleteMem(double**);
```

Loops to delete individual double arrays, then deletes array of pointers to doubles

While loop to go through array of pointers

Delete each array

Delete array of pointers