Lab07

Memory Leaks

- A memory leak occurs when a piece of memory is allocated by the programmer **(usually when using the “new” keyword)** and is then not deallocated
- It’s called a leak because this piece of memory is probably reserved but the program is no longer using it
- To free a single allocated memory space (use the delete keyword): **delete ptr**
- To free an allocated array: **delete [] array;**
- Again the “new” and “delete” keywords are synonymous with dynamic memory allocation (where the size of the required memory is only determined during runtime) like with the heap allocated arrays that we’ve discussed last time

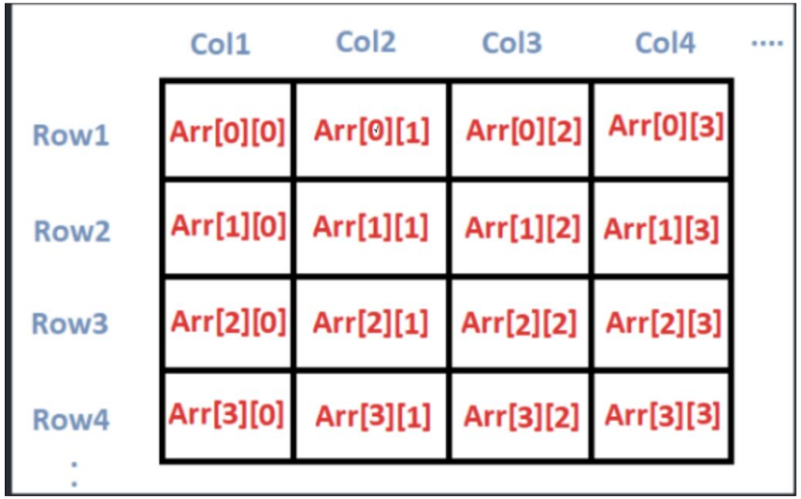
Installing Valgrind for Mac OS

```
brew tap LouisBrunner/valgrind
brew install --HEAD LouisBrunner/valgrind/valgrind
```

```
valgrind --leak-check=full --show-leak-kinds=all
./YourProgram
```

Array (2D array)

- List of one dimensional array.
- 2D array can be thought of as a table or matrix with rows and columns.
  - Eg: int numbers[5][10]. Here 5 represents the row and 10 represents the column.
- Like 1 dimensional array, the very first element of 2D array can be accessed by numbers[0][0] and the last number can be accessed by numbers[row-1][column-1].



	Col1	Col2	Col3	Col4	....
Row1	Arr[0][0]	Arr[0][1]	Arr[0][2]	Arr[0][3]	
Row2	Arr[1][0]	Arr[1][1]	Arr[1][2]	Arr[1][3]	
Row3	Arr[2][0]	Arr[2][1]	Arr[2][2]	Arr[2][3]	
Row4	Arr[3][0]	Arr[3][1]	Arr[3][2]	Arr[3][3]	
:					

Initializing 2D Arrays

```
int** numbers = nullptr;

numbers = new int*[rowCount];     //rowCount=number of rows

for(int i = 0; i < rowCount; i++)

{

numbers[i] = new int[colCount];   //colCount = number of columns

}
```

Deleting Heap Allocated 2D Arrays

```
for(int i = 0; i<rowCount;i++)
{

delete[] numbers[i];
}

delete[] numbers;
```