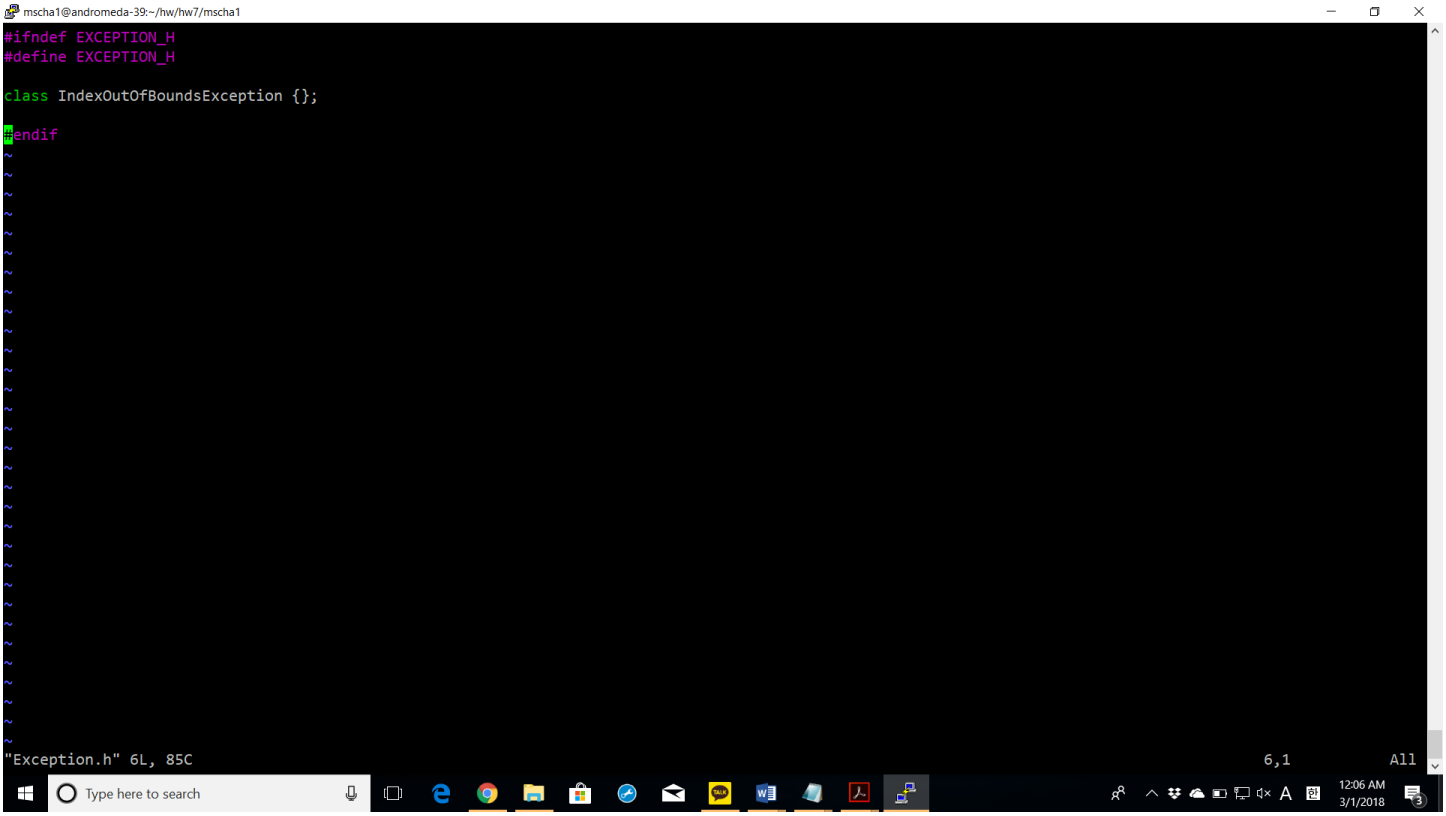
Name: Min Sung Cha Student ID: 85408485

Analysis of functions

Exception.h



The exception header contains the IndexOutOfBoundsException class which is used when the index goes out of bounds for the indexing operator.

Array.h

mscha1@andromeda-39:~/hw/hw7/mscha1

```
#ifndef ARRAY_H
#define ARRAY_H

#include <cassert>
#include <iostream>
#include <iomanip>
#include "Exception.h"

using namespace std;

template
    <typename Type>
class Array
{
    private:
        int len;
        Type * buf;

    public:
        Array(int newLen)
            : len(newLen), buf(new Type[len])
        {
        }

        Array(const Array & l)
            : len(l.len), buf(new Type[l.len])
        {
            for(int i = 0; i < l.len; i++)
                buf[i] = l.buf[i];
        }

        int length(){
            return len;
        }

        Type & operator [] (int i){
            if (i < 0 || i >= len){
"Array.h" 63L, 956C                                                    7,1           Top
```

Array(const Array & l): the copy constructor takes in an array object as a parameter and copies its length and buf into this Array.

mscha1@andromeda-39:~/hw/hw7/mscha1

```
        Type & operator [] (int i){
            if (i < 0 || i >= len){
                throw IndexOutOfBoundsException();
            }
            return buf[i];
        }

        void print(ostream & out){
            for (int i = 0; i < len; ++i)
                out <<setw(8) <<setprecision(2) <<fixed <<right <<buf[i];
        }

        friend ostream & operator << (ostream & out, Array & a){
            a.print(out);
            return out;
        }

        friend ostream & operator << (ostream & out, Array * ap){
            ap -> print(out);
            return out;
        }

        ~Array(){
            delete[] buf;
        }
};

#endif
~
~
~
~
~
~
~
~
~
                                                                       36,2-9        Bot
```

Operator []: if the index supplied as the parameter from the user is smaller than 0 or bigger than or equal to the length, then an IndexOutOfBoundsException is raised. Otherwise, the appropriate element in the index is returned.

Matrix.h

mscha1@andromeda-39:~/hw/hw7/mscha1

```
#ifndef MATRIX_H
#define MATRIX_H
#include "Array.h"
#include "Exception.h"

using namespace std;

template
    <typename Element>
class Matrix
{
    private:
        int rows, cols;

        Array < Array <Element>* > m;

    public:
        Matrix(int newRows, int newCols)
            : rows(newRows), cols(newCols), m(Array<Array<Element>*>(rows))
        {
            for (int i = 0; i < rows; i++)
                m[i] = new Array <Element>(cols);
        }

        int numRows(){
            return rows;
        }

        int numCols(){
           return cols;
        }

        Array <Element> & operator [] (int row)
        {
            if (row < 0 || row >= rows){
                throw IndexOutOfBoundsException();
            }
"Matrix.h" 58L, 904C                                                    4,1           Top
```

Matrix(int newRows, int newCols) – constructs a Matrix objects with the number of rows and columns specified by the user. The constructor defines m, which is an array of arraypointers constructed by calling the Array constructor first on the Array pointer object and then on the type object.

mscha1@andromeda-39:~/hw/hw7/mscha1

```
                m[i] = new Array <Element>(cols);
        }

        int numRows(){
            return rows;
        }

        int numCols(){
           return cols;
        }

        Array <Element> & operator [] (int row)
        {
            if (row < 0 || row >= rows){
                throw IndexOutOfBoundsException();
            }
            return *(m[row]);
        }

        void print(ostream & out){
            for (int i = 0; i < rows; i++)
                out << m[i] << endl;
        }

        friend ostream & operator << (ostream & out, Matrix & m){
            m.print(out);
            return out;
        }

        ~Matrix(){
            for (int i = 0; i < rows; i++){
                delete m[i];
            }
        }
};

#endif
                                                                        57,0-1        Bot
```

~Matrix(): it destructs the Matrix object by deleting every array pointer in every row.

test_matrix.cpp

mscha1@andromeda-39:~/hw/hw7/mscha1

```cpp
#include <iostream>
#include "Matrix.h"
#include "Exception.h"

using namespace std;

template
    <typename T>
void fillMatrix(Matrix <T> & m){
    int i, j;

    for (i = 0; i < m.numRows(); i++)
        m[i][0] = T();
    for (j = 0; j < m.numCols(); j++)
        m[0][j];
    for (i = 1; i < m.numRows(); i++){
        for (j = 1; j < m.numCols(); j++){
            m[i][j] = T(i * j);
        }
    }
}

void test_int_matrix(){
    Matrix <int> m(10, 5);
    fillMatrix(m);
    cout << m;
}

void test_double_matrix(){
    Matrix <double> M(8, 10);
    fillMatrix(M);
    cout << M;
}

void generate_exception(Matrix <double> & m){
    for (int i = 0; i < 666; i++){
        m[i][i] = 10;
"test_matrix.cpp" 64L, 1143C                                    3,1           Top
```

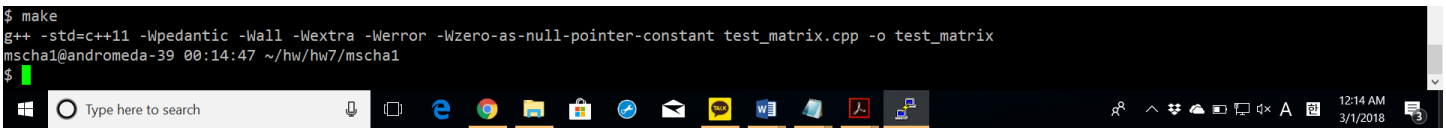
Type here to search 12:07 AM 3/1/2018

mscha1@andromeda-39:~/hw/hw7/mscha1

```cpp

void test_double_matrix(){
    Matrix <double> M(8, 10);
    fillMatrix(M);
    cout << M;
}

void generate_exception(Matrix <double> & m){
    for (int i = 0; i < 666; i++){
        m[i][i] = 10;
    }
}

void test_double_matrix_exceptions(){
    try
    {
        cout << "Starting...\n";
        Matrix <double> M(8, 10);
        fillMatrix(M);
        cout << M;
        generate_exception(M);
        cout << "Done\n";
    }
    catch (IndexOutOfBoundsException & e)
    {
        cout << "Index out of bounds \n";
    }
}

int main(){
    for (int i = 0; i < 3; ++i){
        test_int_matrix();
        test_double_matrix();
        test_double_matrix_exceptions();
    }
    return 0;
}
                                                                28,0-1        Bot
```

Type here to search 12:08 AM 3/1/2018

Screenshots of successful compiling with **make** command



Screenshots of successful execution with **valgrind** command.

mscha1@andromeda-39:~/hw/hw7/mscha1

```
       0       8      16      24      32
       0       9      18      27      36
    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
    0.00    1.00    2.00    3.00    4.00    5.00    6.00    7.00    8.00    9.00
    0.00    2.00    4.00    6.00    8.00   10.00   12.00   14.00   16.00   18.00
    0.00    3.00    6.00    9.00   12.00   15.00   18.00   21.00   24.00   27.00
    0.00    4.00    8.00   12.00   16.00   20.00   24.00   28.00   32.00   36.00
    0.00    5.00   10.00   15.00   20.00   25.00   30.00   35.00   40.00   45.00
    0.00    6.00   12.00   18.00   24.00   30.00   36.00   42.00   48.00   54.00
    0.00    7.00   14.00   21.00   28.00   35.00   42.00   49.00   56.00   63.00
Starting...
    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
    0.00    1.00    2.00    3.00    4.00    5.00    6.00    7.00    8.00    9.00
    0.00    2.00    4.00    6.00    8.00   10.00   12.00   14.00   16.00   18.00
    0.00    3.00    6.00    9.00   12.00   15.00   18.00   21.00   24.00   27.00
    0.00    4.00    8.00   12.00   16.00   20.00   24.00   28.00   32.00   36.00
    0.00    5.00   10.00   15.00   20.00   25.00   30.00   35.00   40.00   45.00
    0.00    6.00   12.00   18.00   24.00   30.00   36.00   42.00   48.00   54.00
    0.00    7.00   14.00   21.00   28.00   35.00   42.00   49.00   56.00   63.00
Index out of bounds
==9179==
==9179== HEAP SUMMARY:
==9179==     in use at exit: 72,704 bytes in 1 blocks
==9179==   total heap usage: 169 allocs, 168 frees, 79,403 bytes allocated
==9179==
==9179== LEAK SUMMARY:
==9179==    definitely lost: 0 bytes in 0 blocks
==9179==    indirectly lost: 0 bytes in 0 blocks
==9179==      possibly lost: 0 bytes in 0 blocks
==9179==    still reachable: 72,704 bytes in 1 blocks
==9179==         suppressed: 0 bytes in 0 blocks
==9179== Rerun with --leak-check=full to see details of leaked memory
==9179==
==9179== For counts of detected and suppressed errors, rerun with: -v
==9179== Use --track-origins=yes to see where uninitialised values come from
==9179== ERROR SUMMARY: 1134 errors from 22 contexts (suppressed: 0 from 0)
mscha1@andromeda-39 00:15:41 ~/hw/hw7/mscha1
$
```

Type here to search

12:16 AM
3/1/2018