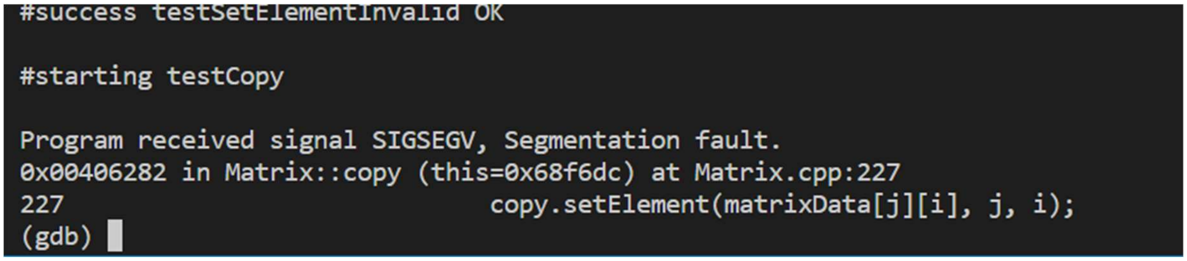
# Debugging Report Lab4

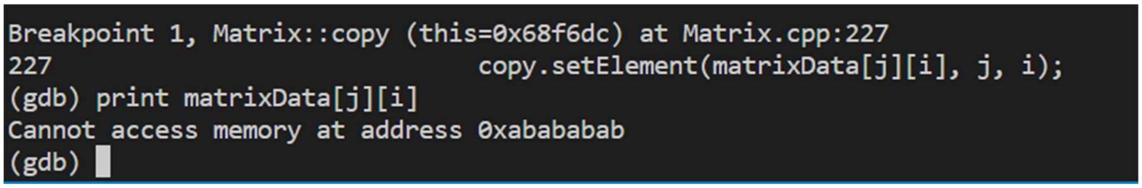


```
#success testSetElementInvalid OK

#starting testCopy

Program received signal SIGSEGV, Segmentation fault.
0x00406282 in Matrix::copy (this=0x68f6dc) at Matrix.cpp:227
227                             copy.setElement(matrixData[j][i], j, i);
(gdb) 
```

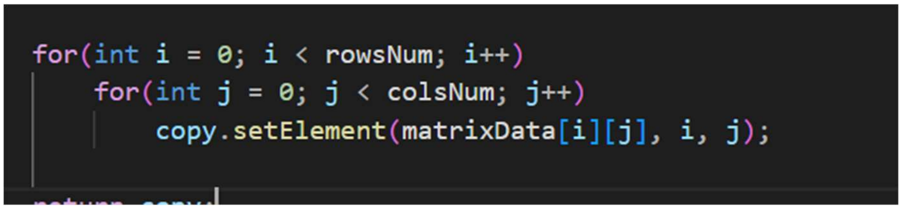
After receiving an error when trying to run the test file, the gdb debugger showed that there was a segmentation fault occurring on line 227. After digging farther, it seems the error was caused by the memory address at matrix Data[j][i] being unaccusable, maybe out of range error?



```
Breakpoint 1, Matrix::copy (this=0x68f6dc) at Matrix.cpp:227
227                             copy.setElement(matrixData[j][i], j, i);
(gdb) print matrixData[j][i]
Cannot access memory at address 0xabababab
(gdb) 
```

It seems that the error was caused by matrixData[j][i] having i, and j in the incorrect spots; proposed fix is to switch them around so that matrixData[j][i] is now matrixData[i][j].

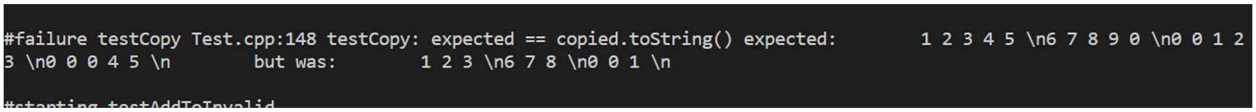
Proposed solution:



```
for(int i = 0; i < rowsNum; i++)
    for(int j = 0; j < colsNum; j++)
        copy.setElement(matrixData[i][j], i, j);

return copy;
```

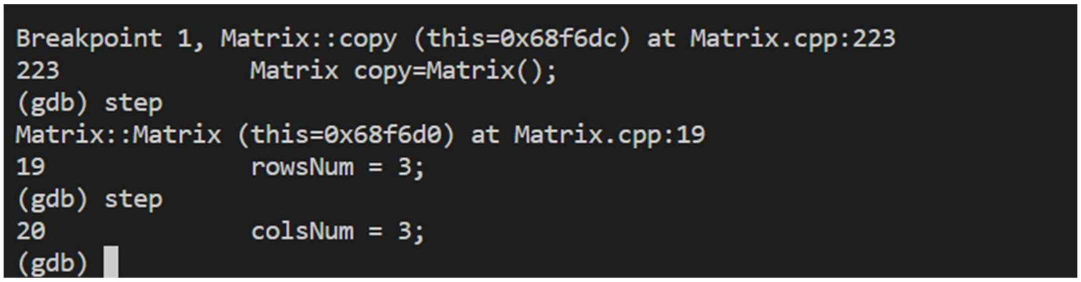
Second issue found is that the rowsNum and colsNum are both equal to 3.



```
#failure testCopy Test.cpp:148 testCopy: expected == copied.toString() expected:       1 2 3 4 5 \n6 7 8 9 0 \n0 0 1 2
3 \n0 0 0 4 5 \n        but was:        1 2 3 \n6 7 8 \n0 0 1 \n

#starting testAddToInvalid
```

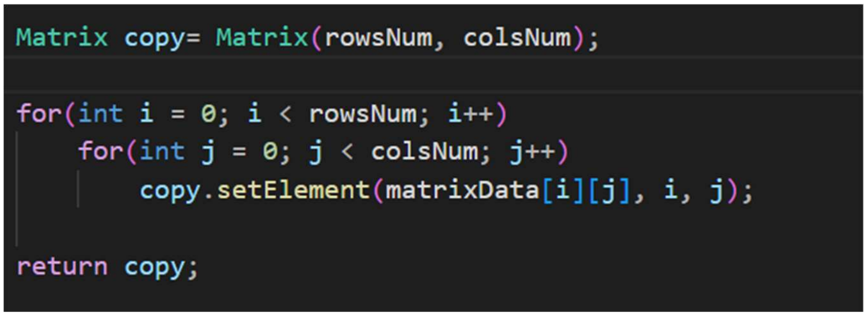
After looking into this deeper it seems that the reason for this is that the Matrix copy initialized before the program is causing the program to enter the constructor Matrix() which initializes rowsNum and colsNum to be 3.



```
Breakpoint 1, Matrix::copy (this=0x68f6dc) at Matrix.cpp:223
223             Matrix copy=Matrix();
(gdb) step
Matrix::Matrix (this=0x68f6d0) at Matrix.cpp:19
19              rowsNum = 3;
(gdb) step
20              colsNum = 3;
(gdb)
```

Proposed solution is to have Matrix copy=Matrix() changed to Matrix copy= Matrix(rowsNum, colsNum). This will have the variables rowsNum and colsNum kept as their intended values and not changed to 3.

Proposed solution:



```
Matrix copy= Matrix(rowsNum, colsNum);

for(int i = 0; i < rowsNum; i++)
    for(int j = 0; j < colsNum; j++)
        copy.setElement(matrixData[i][j], i, j);

return copy;
```