We can start debugging with the first line of the code. Lets put a breakpoint where the matrix is initialized and see if the values are correct:

```
(gdb) b Matrix.cpp: 216
Breakpoint 1 at 0x406267: file Matrix.cpp, line 216.
(gdb) run
```

```
215
216     Matrix copy = Matrix();
217
218     for(int i = 0; i < rowsNum; i++)
219         for(int j = 0; j < colsNum; j++)
220             copy.setElement(matrixData[j][i], j, i);
221
222     return copy;
223
224     /* UNCOMMENT THIS SECTION, THEN DEBUG! */
225     /* UNCOMMENT THIS SECTION, THEN DEBUG! */
226 }
227
228
229 void Matrix::addTo( Matrix m ){
230
231     // Member Function - Add Matrix m to This Matrix
232
```

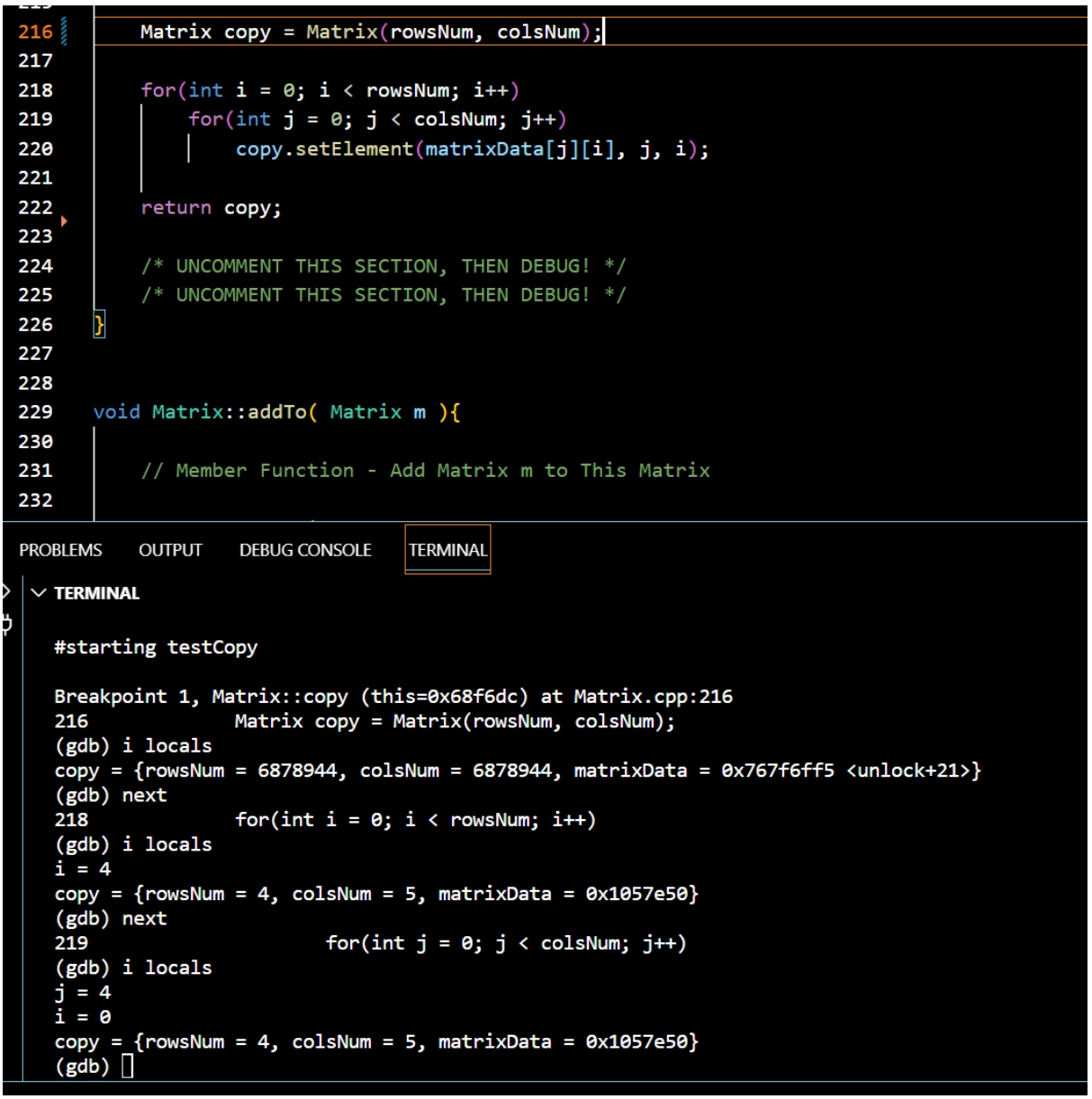
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

TERMINAL

```
Breakpoint 1, Matrix::copy (this=0x68f6dc) at Matrix.cpp:216
216             Matrix copy = Matrix();
(gdb) i locals
copy = {rowsNum = 6878944, colsNum = 6878944, matrixData = 0x767f6ff5 <unlock+21>}
(gdb) next
218             for(int i = 0; i < rowsNum; i++)
(gdb) i locals
i = 4
copy = {rowsNum = 3, colsNum = 3, matrixData = 0xf97e50}
(gdb) next
219                     for(int j = 0; j < colsNum; j++)
(gdb) i locals
j = 4
i = 0
copy = {rowsNum = 3, colsNum = 3, matrixData = 0xf97e50}
(gdb)
```

If we put a breakpoint to line 216 and the type i locals to see the values of the matrix created, we can see that number of rows and columns are set 3 as that is the default size of the matrix when the number or rows and columns are not provided. However we know that the matrix provided can be any size and not always a 3x3 matrix.

This can be fixed this by initializing a matrix with the provided rowsNum and colsNums as can be seen below:



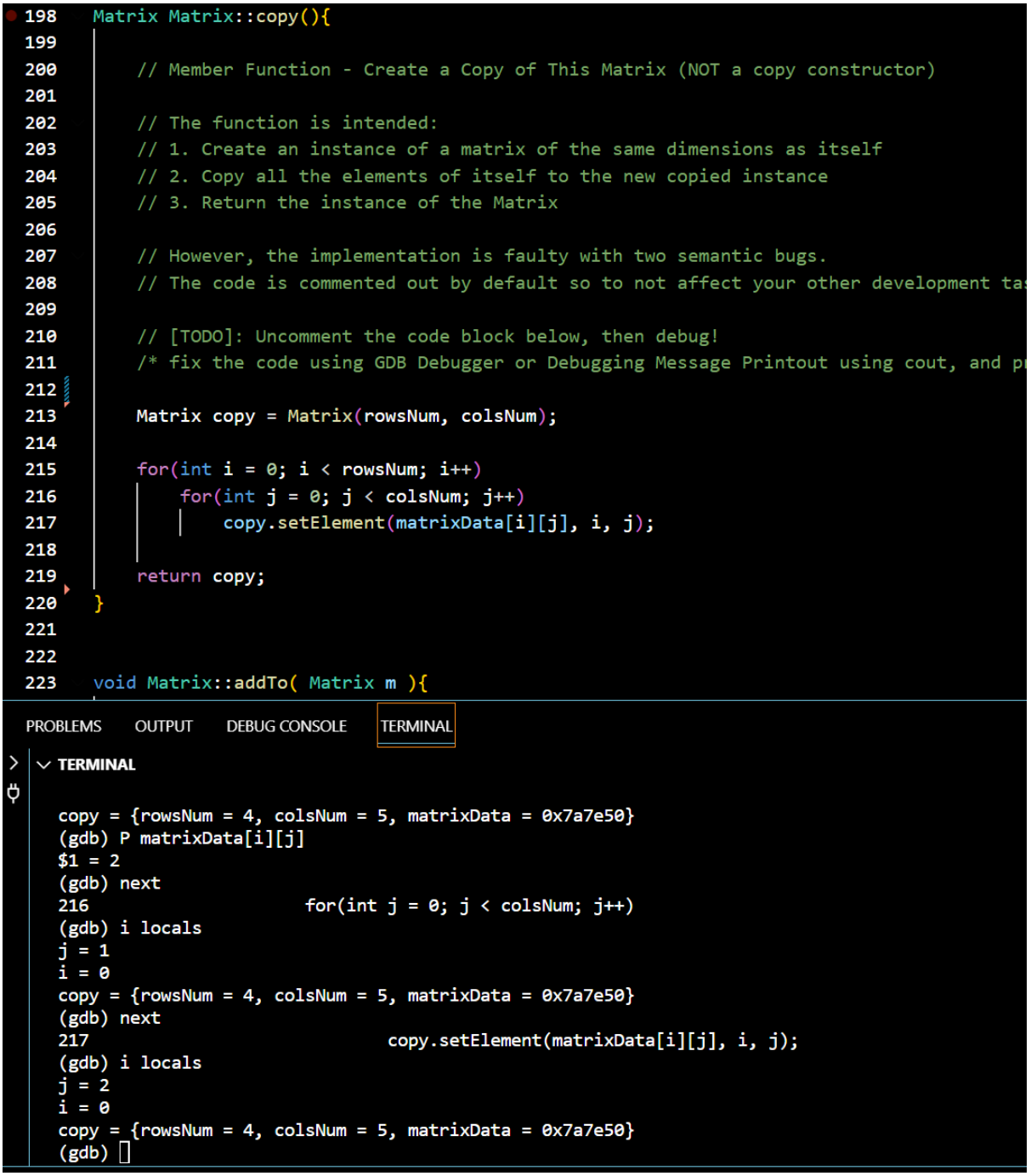
After fixing this, lets keep running the debugger to catch the second error.

```
(gdb) i locals
j = 1
i = 0
copy = {rowsNum = 4, colsNum = 5, matrixData = 0x1057e50}
(gdb) next
220                             copy.setElement(matrixData[j][i], j, i);
(gdb) i locals
j = 2
i = 0
copy = {rowsNum = 4, colsNum = 5, matrixData = 0x1057e50}
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x004062c1 in Matrix::copy (this=0x68f6dc) at Matrix.cpp:220
220                             copy.setElement(matrixData[j][i], j, i);
(gdb) i locals
j = 4
i = 0
copy = {rowsNum = 4, colsNum = 5, matrixData = 0x1057e50}
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x004062c1 in Matrix::copy (this=0x68f6dc) at Matrix.cpp:220
220                             copy.setElement(matrixData[j][i], j, i);
(gdb)
```

We see that the elements are not being added correctly and we receive segmentation faults. This is due to the indices j & i being added to the new matrix in the wrong order. Lets write them in the correct order and rerun the debugger:



```
198  Matrix Matrix::copy(){
199
200      // Member Function - Create a Copy of This Matrix (NOT a copy constructor)
201
202      // The function is intended:
203      // 1. Create an instance of a matrix of the same dimensions as itself
204      // 2. Copy all the elements of itself to the new copied instance
205      // 3. Return the instance of the Matrix
206
207      // However, the implementation is faulty with two semantic bugs.
208      // The code is commented out by default so to not affect your other development ta
209
210      // [TODO]: Uncomment the code block below, then debug!
211      /* fix the code using GDB Debugger or Debugging Message Printout using cout, and p
212
213      Matrix copy = Matrix(rowsNum, colsNum);
214
215      for(int i = 0; i < rowsNum; i++)
216          for(int j = 0; j < colsNum; j++)
217              copy.setElement(matrixData[i][j], i, j);
218
219      return copy;
220  }
221
222
223  void Matrix::addTo( Matrix m ){
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

TERMINAL

```
copy = {rowsNum = 4, colsNum = 5, matrixData = 0x7a7e50}
(gdb) P matrixData[i][j]
$1 = 2
(gdb) next
216                     for(int j = 0; j < colsNum; j++)
(gdb) i locals
j = 1
i = 0
copy = {rowsNum = 4, colsNum = 5, matrixData = 0x7a7e50}
(gdb) next
217                             copy.setElement(matrixData[i][j], i, j);
(gdb) i locals
j = 2
i = 0
copy = {rowsNum = 4, colsNum = 5, matrixData = 0x7a7e50}
(gdb)
```

Code is now fixed!