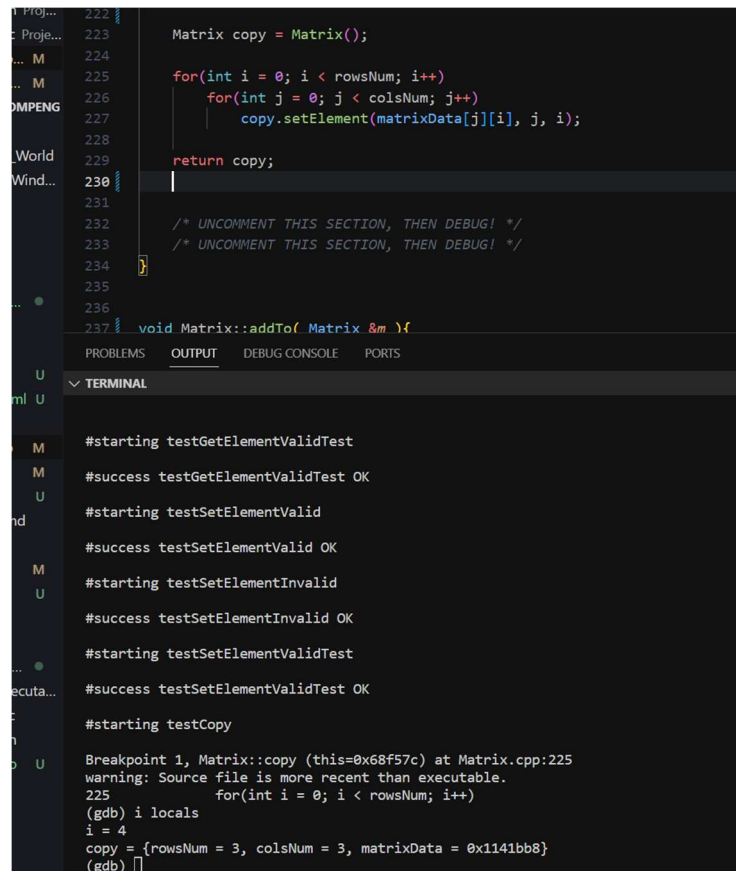


Debugging Report

Vraj Patel
400434795
patev61

Bug #1

Indetification of Bug:



The screenshot shows a code editor with a C++ file. The code defines a Matrix class with a copy constructor. The copy constructor initializes a new Matrix object and then copies the data from the original Matrix. The terminal window shows the output of the tests, including the success of the copy test. The terminal output is as follows:

```
#starting testGetElementValidTest
#success testGetElementValidTest OK
#starting testSetElementValid
#success testSetElementValid OK
#starting testSetElementInvalid
#success testSetElementInvalid OK
#starting testSetElementValidTest
#success testSetElementValidTest OK
#starting testCopy
Breakpoint 1, Matrix::copy (this=0x68f57c) at Matrix.cpp:225
warning: Source file is more recent than executable.
225     for(int i = 0; i < rowsNum; i++)
(gdb) i locals
i = 4
copy = {rowsNum = 3, colsNum = 3, matrixData = 0x1141bb8}
(gdb) []
```

- In the picture above, we see the copy Matrix which consists of 3 as its rowsNum and colsNum which is automativally given to the copy Matrix through the default constructor
- In Test.cpp, the testcase for copy shows us that rowsNum and colsNum should be 4,5 respectively, but in this case it is 3,3

```
void testCopy() {  
  
    int row=4,col=5;  
    int data[4][5] = {{1, 2, 3, 4, 5}, {6, 7, 8, 9, 0}, {0, 0, 1, 2, 3}, {0, 0, 0, 4, 5}};
```

Debugging Report

Vraj Patel
400434795
patev61

Proposal of Bug Fix:

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

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

return copy;
```

- The root of the bug is from the copy Matrix being ran through the default constructor, it automatically initializes the rowsNum and colsNum
- To fix the bug, passing through rowsNum and colsNum as the parameter will run through the additional Constructor which initializes rowsNum to rows and colsNum to cols
- This means that rowsNum would be 4 and colsNum would be 5, as expected from the testcase

Validation of Bug Fix:

```
Breakpoint 1, Matrix::copy (this=0x68f57c) at Matrix.cpp:225
225         for(int i = 0; i < rowsNum; i++)
(gdb) i locals
i = 4
copy = {rowsNum = 4, colsNum = 5, matrixData = 0x10e1bb8}
(gdb) █
```

- This fixed the bug and initialized the copy to 4 and 5 which we want according to our test case

Debugging Report

Vraj Patel
400434795
patev61

Bug #2

Identification of Bug:

```
(gdb) s
Matrix::copy (this=0x68f57c) at Matrix.cpp:226
226             for(int j = 0; j < colsNum; j++)
(gdb) s
227             copy.setElement(matrixData[j][i], j, i);
(gdb) i locals
j = 4
i = 0
copy = {rowsNum = 4, colsNum = 5, matrixData = 0x1091bb8}
(gdb) s

Program received signal SIGSEGV, Segmentation fault.
0x0040833d in Matrix::copy (this=0x68f57c) at Matrix.cpp:227
```

- After stepping through the break set up at line 225, there is a Segmentation Fault Error
- After further observation, this error is caused by out of scope on the copy.setElement line of the code
- J = 4, but since index start from 0 in C++, the maxIndex that j can be is 3 (since rowsNum is 4), J is 4 in this case (i locals) which causes an Segmentation Fault since the program is trying to access something which is "out of bounds"
-

Proposal of Bug Fix:

```
21
22
23 Matrix copy = Matrix(rowsNum, colsNum);
24
25 for(int i = 0; i < rowsNum; i++)
26     for(int j = 0; j < colsNum; j++)
27         copy.setElement(matrixData[i][j], i, j);
28
29 return copy;
30
31
32 /* UNCOMMENT THIS SECTION, THEN DEBUG! */
33 /* UNCOMMENT THIS SECTION, THEN DEBUG! */
```

- In the code above index I and J were flipped which ensures that Segmentation fault doesn't occur
- In the for loop i is iterated to rowsNum-1 and j is iterated to colsNum-1 and we know that in this case that the first square brackets of matrixData refers to rows and the second, columns, so the new code matches that phenomenon as well

Debugging Report

Vraj Patel
400434795
patev61

Validation of Bug Fix:

```
(gdb) s
Matrix::copy (this=0x68f52c) at Matrix.cpp:226
226             for(int j = 0; j < colsNum; j++)
(gdb) i locals
j = 4
i = 0
copy = {rowsNum = 4, colsNum = 5, matrixData = 0x1091c18}
(gdb) s
225             for(int i = 0; i < rowsNum; i++)
(gdb) s
226             for(int j = 0; j < colsNum; j++)
(gdb) s
227             copy.setElement(matrixData[i][j], i, j);
(gdb) i locals
j = 0
i = 1
copy = {rowsNum = 4, colsNum = 5, matrixData = 0x1091c18}
(gdb) |
```

- In the picture above we can see that j goes to its new maxIndex which is 4 and then the program goes out to the outer for loop, iterating i=1
- The code successfully identifies the right maxIndex and prevents the segmentation Fault allowing it to successfully pass the testcase for copy

```
#success testSetElementValidTest OK

#starting testCopy

#success testCopy OK

#starting testAddToInvalid

#success testAddToInvalid OK
```