Lab4 Matrix::copy() Debugging Report | xuej41 | 400515671

Bug #1 – Rows and Columns are swapped, causing segmentation fault

```
Matrix copy = Matrix();

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

return copy;
```

I am using cout statements to see what is going wrong. I have underlined the cout line I added in red. This should show the elements of matrixData that are being copied into the copy matrix.

```
=== testCopy() ===
1
6
0
0
Segmentation fault
```

```
string row1 = "1 2 3 4 5 \n", row2 = "6 7 8 9 0 \n", row3 = "0 0 1 2 3 \n", row4 = "0 0 0 4 5 \n";
```

However, there is a segmentation fault. This is because i and j are swapped. j is apart of the nested for loop in charge of the columns, put is placed first in matrixData. So it gets the first element of each row in row1, which is 1, 6, 0, and 0. It then attempts to get the 5th element, but cannot as there are only 4 rows, causing a segmentation fault.

A simple fix is to swap the i and j.

The new line should read **copy.setElement(matrixData[i][j], i, j);**

Bug #2 – Rows and Columns are not defined, causing them to default to 3

```
Matrix copy = Matrix();

cout<<"Rows: "<<rowsNum<<endl;
cout<<"Colmns: "<<colsNum<<endl;

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

return copy;
```

As you can see, when the copy matrix is created, it is given no arguments, causing it to be created with the default constructor which assumes 3 rows and 3 columns.
I have circled the issue and drawn a red rectangle around the cout statements I used to catch this error.
Referencing the test matrix again:

```
string row1 = "1 2 3 4 5 \n", row2 = "6 7 8 9 0 \n", row3 = "0 0 1 2 3 \n", row4 = "0 0 0 4 5 \n";
```

This matrix has 4 rows and 5 columns. The copy matrix therefore won't be getting every element copied, only the first 3 rows and 3 columns. Running the executable confirms this.

```
=== testCopy() ===
Rows: 4
Colmns: 5
                [ASSERTION] Expected: 1 2 3 4 5
6 7 8 9 0
0 0 1 2 3
0 0 0 4 5
, but Actual: 1 2 3
6 7 8
0 0 1

!!! FAILED !!!
```

My cout commands confirm that the test matrix has 4 Rows and 5 Columns. And as predicted, it expected the copy matrix to be 12345, 67890 etc. But it only became 123, 678 etc.
A simple fix is to define the rows and columns for the copy matrix, so that the other constructor is used.

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

cout<<"Rows: "<<rowsNum<<endl;
cout<<"Colmns: "<<colsNum<<endl;

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

return copy;
```

After adding this fix, the copy matrix works.

```
=== testCopy() ===
Rows: 4
Colmns: 5
Passed
```