We keep working with the Graph class defined in the previous exercise sheets. This time we are going to assume that we want to represent sparse graphs (notice that an adjacency matrix would not be the best data structure in this case)

1. Add a copy constructor and a copy assignment operator to your graph. Show an example of using it.

2. Add a move constructor and a move assignment operator to your graph. Shown an example of using it.

3. Modify your class so it does not allow a graph to be passed as value to a function.

4. Consider the Spreadsheet example examined during the lecture:

```
class SpreadsheetCell {
public:
     double value;
};

class Spreadsheet {
public:
     Spreadsheet(int inWidth, int inHeight);
     Spreadsheet(const Spreadsheet &src);
     Spreadsheet& operator=(const Spreadsheet & rhs);
     Spreadsheet(Spreadsheet &&src) noexcept;
     Spreadsheet & operator=(Spreadsheet &&rhs) noexcept;
private:
     int mWidth, mHeight;
     SpreadsheetCell** mCells;
};
```

Provide your own implementation of these constructors based on the given data structure. Once implemented execute the following piece of code:

```
Spreadsheet CreateObject() {
     return Spreadsheet(3, 2);
}
int main() {
     std::vector<Spreadsheet> vec;
     for (auto i = 0; i < 2; ++i)
          vec.push_back(Spreadsheet(100, 100));


     Spreadsheet s(2, 3);
     s = CreateObject();
     Spreadsheet s2(5, 6);
     s2 = s;
     Spreadsheet s3 = s;
}
```

How many times each constructor is called and in which order.