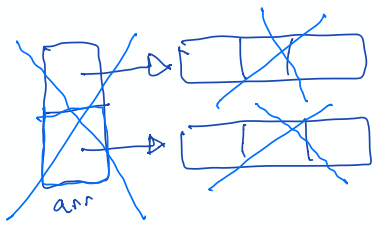
Create a Class called Matrix that owns a row × column two-dimensional array on the freestore. Implement the constructor, destructor, copy constrictor, copy assignment operator for this type.

```
template <typename T>
class Matrix {
public:
    Matrix(int, int);
    Matrix(const Matrix<T>&);
    ~Matrix();
    Matrix<T>& operator=(const Matrix<T>&);

private:
    int rows;
    int cols;
    T **arr;
};
```

```
template <typename T>
Matrix<T>::Matrix(int r, int c) : row(r), col(c), arr(nullptr)
{
    arr = new T*[r];
    for (int i=0; i<row; ++i)
        arr[i] = new T[c];
}
```




```
template <typename T>
Matrix<T>::~Matrix()
{
    for (int i=0; i<row; ++i)
        delete [] arr[i];
    delete[] arr;
}
```

```
template <typename T>
Matrix<T>::Matrix(const Matrix<T> &rhs) :
    row(rhs.row), col(rhs.col),
    arr(nullptr)
{
    arr = new T*[row];
    for (int i=0; i<row; ++i)
    {
        arr[i] = new T[col];
        for (int j=0; j<col; ++j)
            arr[i][j] = rhs.arr[i][j];
    }
}
```

```
template <typename T>
Matrix<T>& Matrix<T>::operator=(const Matrix<T>& rhs)
{
    if (this == &rhs)
        return *this;

    Matrix<T> temp(rhs);
    std::swap(row, temp.row);
    std::swap(col, temp.col);
    std::swap(arr, temp.arr);

    return *this;
}
```

```
Matrix<int> m1, m2, m3;
// ... do something
m1 = m2 = m3;
```