Course: Programming Fundamental – ENSF 337

Lab #: 9

Instructor: Khedr

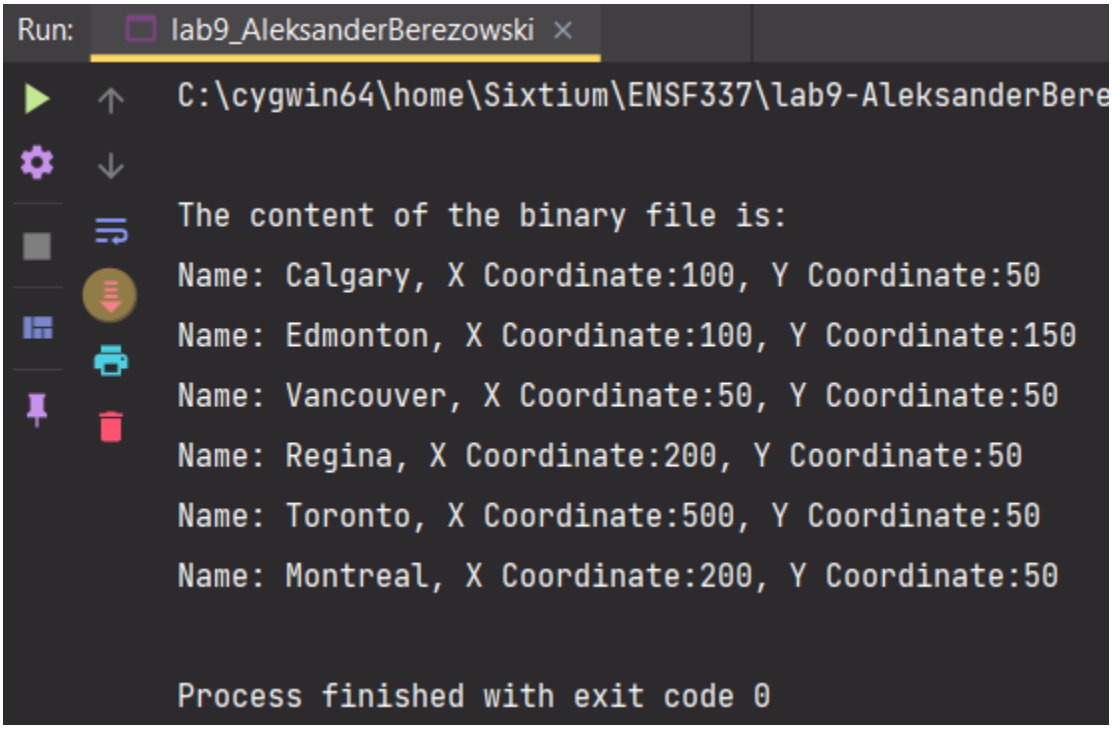
Student Name: Aleksander Berezowski

Lab Section: B04

Date submitted: On or before December 3rd

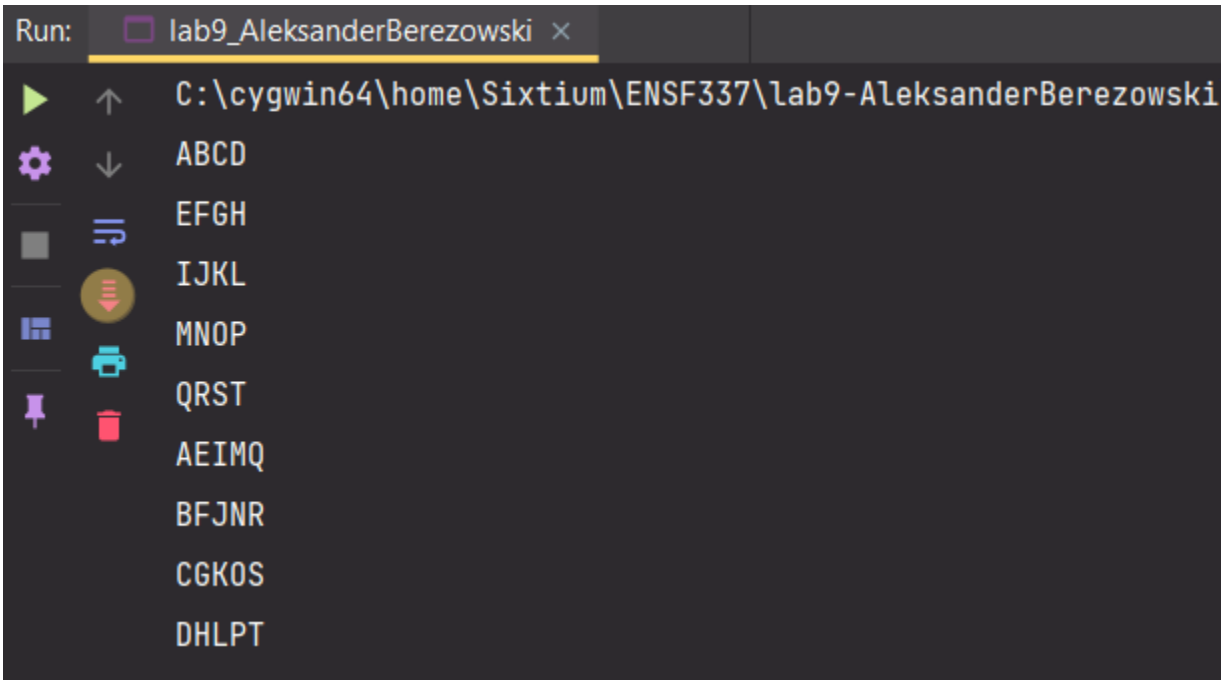
Exercise B

```
void print_from_binary(char* filename) {
    ifstream stream(filename, ios::out | ios::binary);
    if(stream.fail()){
        cout << filename << " failed to open" << endl;
        exit(1);
    }
    City cityObjects [1000];
    stream.seekg(0L, ios::end);
    int amount = stream.tellg()/sizeof(City);
    stream.seekg(0L, ios::beg);
    stream.read((char*)cityObjects, sizeof(City) * amount);
    stream.close();

    int i = 0;
    while (i<amount){
        cout << "Name: " << cityObjects[i].name;
        cout << ", X Coordinate:" << cityObjects[i].x;
        cout << ", Y Coordinate:" << cityObjects[i].y << endl;
        i++;
    }
}
```

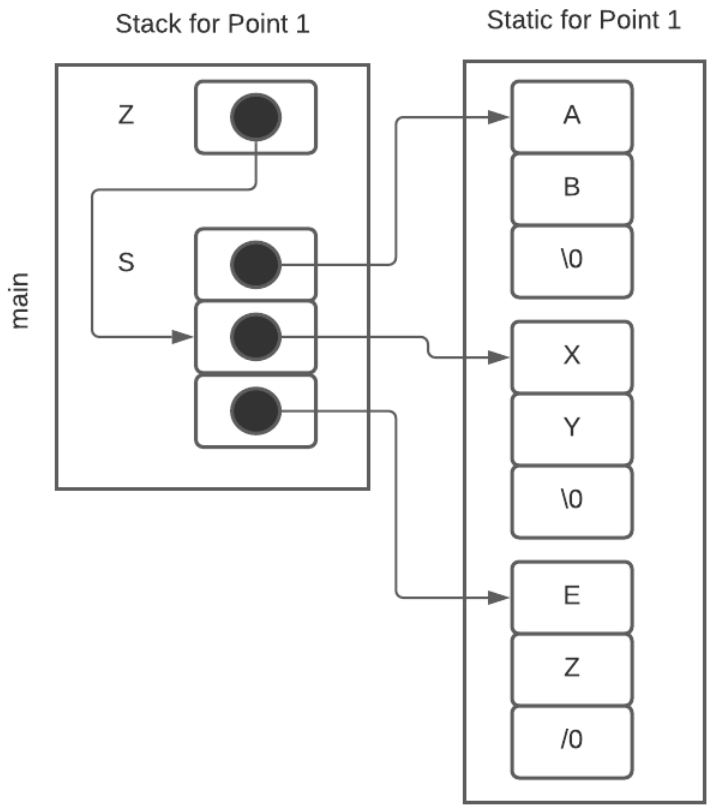


Exercise C

```
String_Vector transpose (const String_Vector& sv) {
    String_Vector vs;
    vs.resize(sv[0].size());
    int x, y;

    for (x = 0; x < sv[x].size(); x++){
        for (y = 0; y < sv.size(); y++){
            vs[x].push_back(sv[y][x]);
        }
    }
    return vs;
}
```



Exercise D



Predicted Output:

The value of **z is: X

The value of *z is: XY

The value of **(z-1) is: A

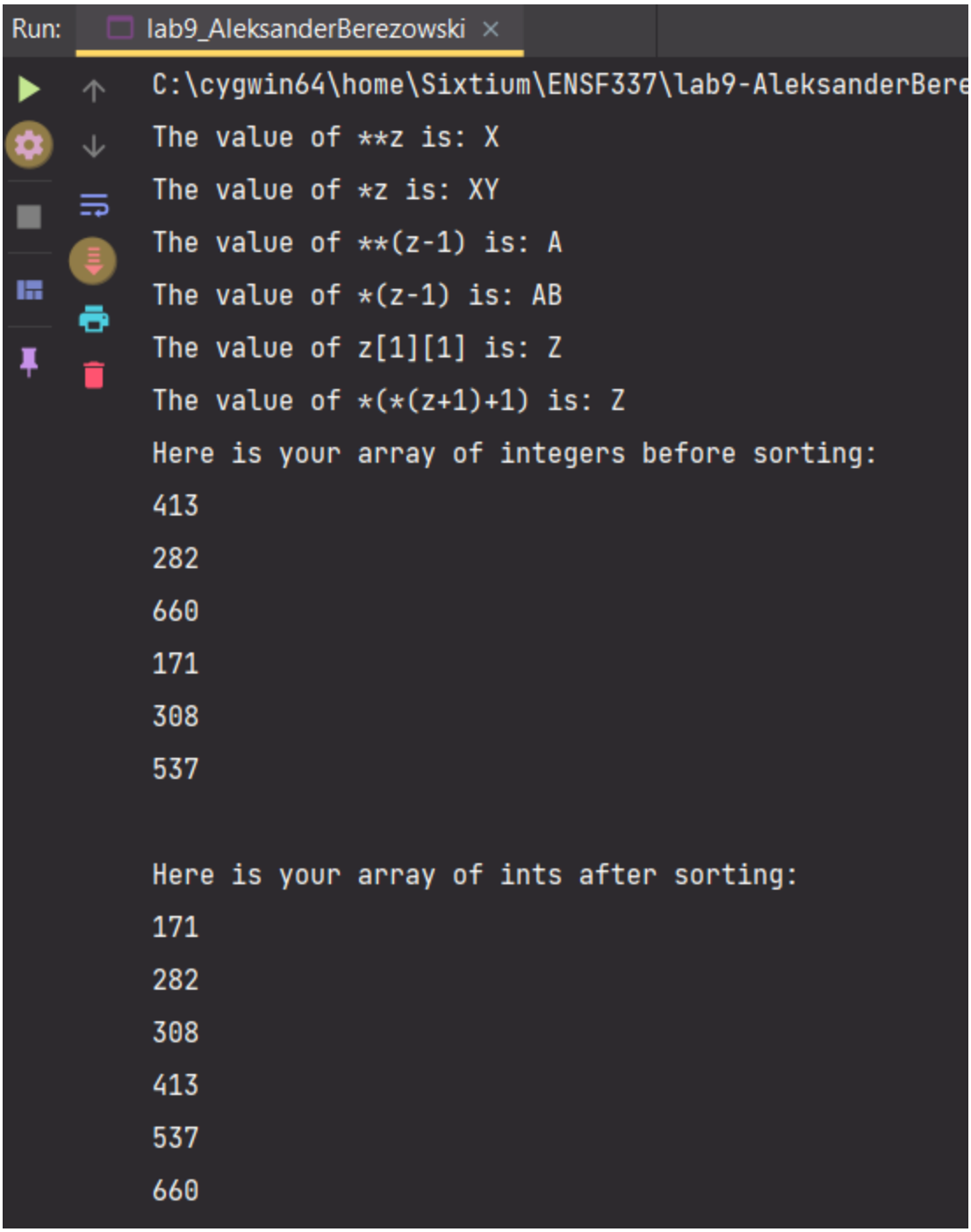
The value of *(z-1) is: AB

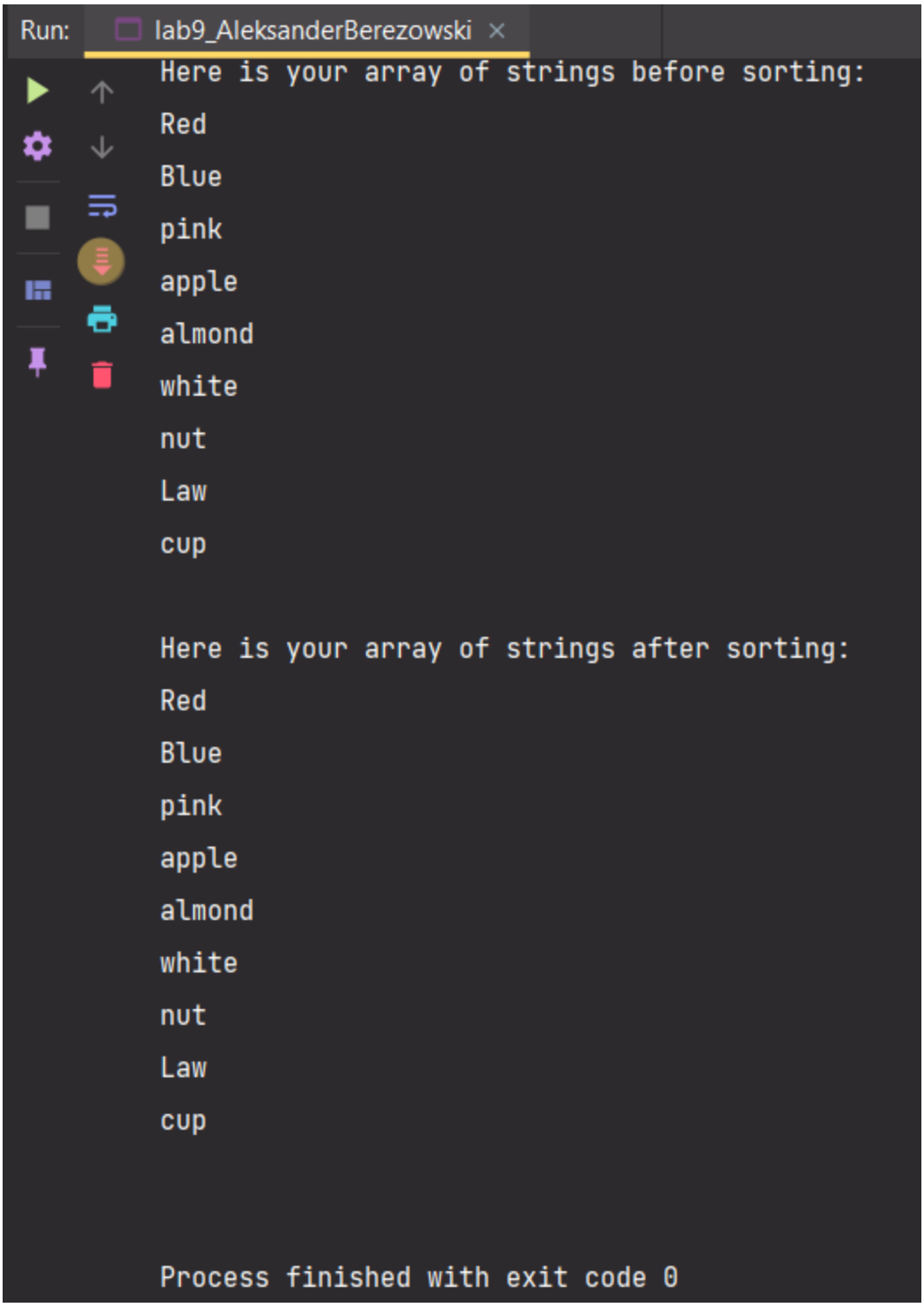
The value of z[1][1] is: Z

The value of *(*(z+1)+1) is: Z

```cpp
#include <iostream>
using namespace std;

void insertion_sort(int *int_array, int n);
/* REQUIRES
 *    n > 0.
 *    Array elements int_array[0] ... int_array[n - 1] exist.
 * PROMISES
 *    Element values are rearranged in non-decreasing order.
 */

void insertion_sort(const char** str_array, int n);

/* REQUIRES
 *   n > 0.
 *   Array elements str_array[0] ... str_array[n - 1] exist.
 * PROMISES
 *   pointers in str_array are rearranged so that strings:
 *   str_array[0] points to a string with the smallest string (lexicographicall) ,
 *   str_array[1] points to the second smallest string, ..., str_array[n-2]
 *   points to the second largest, and str_array[n-1] points to the largest string
 */

int main(void)
{
    const char* s[] = { "AB", "XY", "EZ"};
    const char** z = s;
    z += 1;


    cout << "The value of **z is: " << **z << endl;
    cout << "The value of *z is: " << *z << endl;
    cout << "The value of **(z-1) is: " << **(z-1)<< endl;
    cout << "The value of *(z-1) is: " << *(z-1)<< endl;
    cout << "The value of z[1][1] is: " << z[1][1]<< endl;
    cout << "The value of *(*(z+1)+1) is: " << *(*(z+1)+1)<< endl;

    // point 1

    int a[] = { 413, 282, 660, 171, 308, 537 };

    int i;
    int n_elements = sizeof(a) / sizeof(int);

    cout << "Here is your array of integers before sorting: \n";
    for(i = 0; i < n_elements; i++)
        cout <<  a[i] << endl;
    cout << endl;

    insertion_sort(a, n_elements);

    cout << "Here is your array of ints after sorting:  \n" ;
    for(i = 0; i < n_elements; i++)
        cout << a[i] << endl;
#if 1
    const char* strings[] = { "Red", "Blue", "pink","apple", "almond","white",
                                            "nut", "Law", "cup"};

    n_elements = sizeof(strings) / sizeof(char*);

    cout << "\nHere is your array of strings before sorting: \n";
    for(i = 0; i < n_elements; i++)
        cout <<  strings[i] << endl;
```

```
    cout << endl;

    insertion_sort(strings, 9);


    cout << "Here is your array of strings after sorting:  \n" ;
    for(i = 0; i < n_elements; i++)
        cout << strings[i] << endl;
    cout << endl;

#endif

    return 0;
}

void insertion_sort(int *a, int n)
{
    int i;
    int j;
    int value_to_insert;

    for (i = 1; i < n; i++) {
        value_to_insert = a[i];

        /* Shift values greater than value_to_insert. */
        j = i;
        while ( j > 0 && a[j - 1] > value_to_insert  ) {
            a[j] = a[j - 1];
            j--;
        }

        a[j] = value_to_insert;
    }
}


void insertion_sort(const char** str_array, int n){
    int i, j;
    const char *t;
    for (i = 0; i < n; i++) {
        for (j = i + 1; j < n; j++) {
            if (*(str_array + j) < *(str_array + i)) {
                t = *(str_array + i);
                *(str_array + i) = *(str_array + j);
                *(str_array + j) = t;
            } else if (*(str_array + j)[0] == *(str_array + i)[0]) {
                if  (*(str_array + j)+1 < *(str_array + i)+1) {
                    t = *(str_array + i);
                    *(str_array + i) = *(str_array + j);
                    *(str_array + j) = t;
                }
            }
        }
    }

}
```


Run:
lab9_AleksanderBerezowski
C:\cygwin64\home\Sixtium\ENSF337\lab9-AleksanderBere
The value of **z is: X
The value of *z is: XY
The value of **(z-1) is: A
The value of *(z-1) is: AB
The value of z[1][1] is: Z
The value of *(*(z+1)+1) is: Z
Here is your array of integers before sorting:
413
282
660
171
308
537
Here is your array of ints after sorting:
171
282
308
413
537
660


Run:
lab9_AleksanderBerezowski
Here is your array of strings before sorting:
Red
Blue
pink
apple
almond
white
nut
Law
cup
Here is your array of strings after sorting:
Red
Blue
pink
apple
almond
white
nut
Law
cup
Process finished with exit code 0

Exercise E

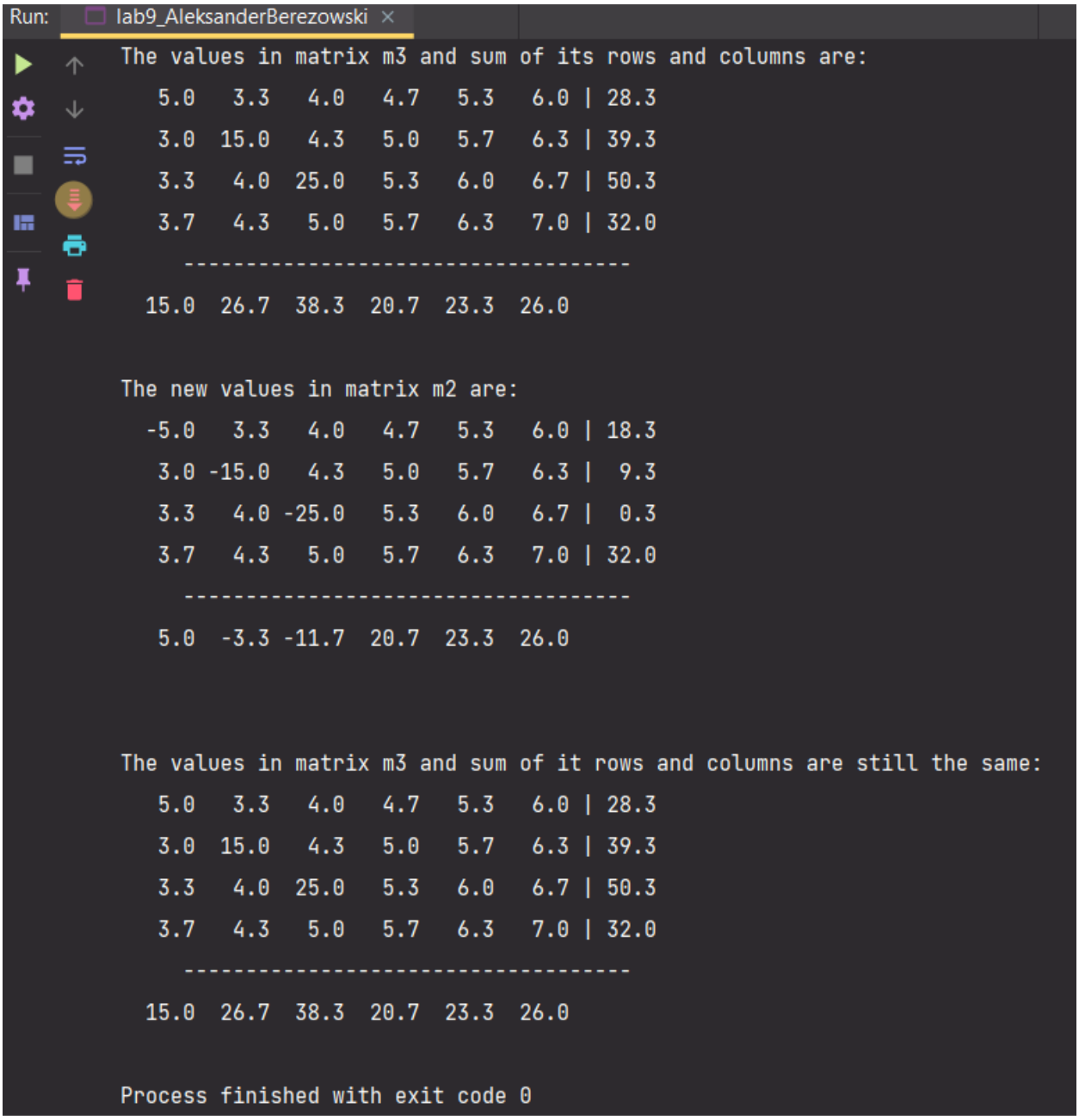
```
//   matrix.cpp


#include "matrix.h"

Matrix::Matrix(int r, int c):rowsM(r), colsM(c)
{
    matrixM = new double* [rowsM];
    assert(matrixM != NULL);

    for(int i=0; i < rowsM; i++){
        matrixM[i] = new double[colsM];
        assert(matrixM[i] != NULL);
    }
    sum_rowsM = new double[rowsM];
    assert(sum_rowsM != NULL);

    sum_colsM = new double[colsM];
    assert(sum_colsM != NULL);
}


Matrix::~Matrix()
{
    destroy();
}

Matrix::Matrix(const Matrix& source)
{
    copy(source);
}

Matrix& Matrix::operator= (const Matrix& rhs)
{
    if(&rhs != this){
        destroy();
        copy(rhs);
    }

    return *this;
}

double Matrix::get_sum_col(int i) const
{
    assert(i >= 0 && i < colsM);
    return sum_colsM[i];
}

double Matrix::get_sum_row(int i) const
{
    assert(i >= 0 && i < rowsM);
    return sum_rowsM[i];
}


void Matrix::sum_of_rows()const
{
    double sum;
     for(int i = 0; i<rowsM; i++) {
        sum = 0;
         for (int j = 0; j < colsM; j++) {
```

```
            sum += matrixM[i][j];
        }
         sum_rowsM[i] = sum;
    }
}

void Matrix::sum_of_cols()const
{
    double sum;
    for(int i = 0; i<colsM; i++) {
        sum = 0;
        for (int j = 0; j < rowsM; j++) {
            sum += matrixM[j][i];
        }
        sum_colsM[i] = sum;
    }
}

void Matrix::copy(const Matrix& source)
{
    // THIS FUNCITON IS DEFECTIVE AND DOSEN'T PROPERLY MAKE THE COPY OF SROUCE
    if(source.matrixM == NULL){
        matrixM = NULL;
        sum_rowsM = NULL;
        sum_colsM = NULL;
        rowsM = 0;
        colsM = 0;
        return;
    }

    rowsM = source.rowsM;
    colsM = source.colsM;

    sum_rowsM = new double[rowsM];
    assert(sum_rowsM != NULL);


    sum_colsM = new double[colsM];
    assert(sum_colsM != NULL);

    matrixM = new double*[rowsM];
    assert(matrixM !=NULL);
    for(int i =0; i < rowsM; i++){
        matrixM[i] = new double[colsM];
        assert(matrixM[i] != NULL);
    }
    for (int i = 0; i < rowsM; i++) {
        for (int j = 0; j < colsM; ++j) {
            matrixM[i][j] = source.at(i,j);
        }
        sum_rowsM[i] = source.sum_rowsM[i];
    }
    for (int i = 0; i < colsM; ++i) {
        sum_colsM[i] = source.sum_colsM[i];
    }
}



void Matrix::destroy()
{
    for(int i = 0; i<rowsM; i++) {
        delete matrixM[i];
```

```
    }
    delete matrixM;
    delete sum_colsM;
    delete sum_rowsM;
}
```

Run: lab9_AleksanderBerezowski ×

```
C:\cygwin64\home\Sixtium\ENSF337\lab9-AleksanderBerezowski\cmake-


The values in matrix m1 are:

   2.3   3.0   3.7   4.3
   2.7   3.3   4.0   4.7
   3.0   3.7   4.3   5.0


The values in matrix m2 are:
   2.7   3.3   4.0   4.7   5.3   6.0
   3.0   3.7   4.3   5.0   5.7   6.3
   3.3   4.0   4.7   5.3   6.0   6.7
   3.7   4.3   5.0   5.7   6.3   7.0


The new values in matrix m1 and sum of its rows and columns are
   2.7   3.3   4.0   4.7   5.3   6.0 | 26.0
   3.0   3.7   4.3   5.0   5.7   6.3 | 28.0
   3.3   4.0   4.7   5.3   6.0   6.7 | 30.0
   3.7   4.3   5.0   5.7   6.3   7.0 | 32.0
     -----------------------------------
  12.7  15.3  18.0  20.7  23.3  26.0
```


Run:
lab9_AleksanderBerezowski
The values in matrix m3 and sum of its rows and columns are:
5.0 3.3 4.0 4.7 5.3 6.0 | 28.3
3.0 15.0 4.3 5.0 5.7 6.3 | 39.3
3.3 4.0 25.0 5.3 6.0 6.7 | 50.3
3.7 4.3 5.0 5.7 6.3 7.0 | 32.0
15.0 26.7 38.3 20.7 23.3 26.0
The new values in matrix m2 are:
-5.0 3.3 4.0 4.7 5.3 6.0 | 18.3
3.0 -15.0 4.3 5.0 5.7 6.3 | 9.3
3.3 4.0 -25.0 5.3 6.0 6.7 | 0.3
3.7 4.3 5.0 5.7 6.3 7.0 | 32.0
5.0 -3.3 -11.7 20.7 23.3 26.0
The values in matrix m3 and sum of it rows and columns are still the same:
5.0 3.3 4.0 4.7 5.3 6.0 | 28.3
3.0 15.0 4.3 5.0 5.7 6.3 | 39.3
3.3 4.0 25.0 5.3 6.0 6.7 | 50.3
3.7 4.3 5.0 5.7 6.3 7.0 | 32.0
15.0 26.7 38.3 20.7 23.3 26.0
Process finished with exit code 0