Ranoa, Jason
CSC 121 001 Computer Science I
Midterm Part II Submission

Screenshot:

```
0 1 2 3 4 5 6 7 8 9
Average: 4.5
Process finished with exit code 0
```

Output: *sorted.txt*
Contains (copied directly): `0 1 2 3 4 5 6 7 8 9`

Source Files:

1. *main.cpp*
2. *ArrayUtility.h*
3. *ArrayUtility.cpp*
4. Object Folder > *unsorted.txt*

Source Code:

1. **main.cpp**

```cpp
#include <iostream>
#include "ArrayUtility.h"
using namespace std;

void doStuff();

int main() {
    doStuff();
    return 0;
}

void doStuff() {
    ArrayUtility au;
    const int SIZE = 10;
    int numbers[SIZE];

    if (!au.readNumbers(numbers, SIZE)) {
        cout << "Cannot read file. "
             << "Something went wrong." << endl;
        return;
    }

    au.selectionSort(numbers, SIZE);
    au.displayNumbers(numbers, SIZE);

    if (!au.writeNumbers(numbers, SIZE)) {
        cout << "Cannot write file. "
             << "Something went wrong." << endl;
        return;
    }

    cout << endl;
    cout << "Average: " << au.calcAvg(numbers, SIZE);
}
```

2. **ArrayUtility.h**

```h
#ifndef MIDTERM_ARRAYUTILITY_H
#define MIDTERM_ARRAYUTILITY_H

class ArrayUtility {

public:
    bool readNumbers(int[], int);
    void displayNumbers(int[], int);
    void selectionSort(int[], int);
    bool writeNumbers(int[], int);
    double calcAvg(int[], int);
};

#endif //MIDTERM_ARRAYUTILITY_H
```

3. **ArrayUtility.cpp**

```cpp
#include "ArrayUtility.h"
#include <iostream>
#include <fstream>
using namespace std;

bool ArrayUtility::readNumbers(int nlist[], int size) {
    ifstream file;

    file.open("unsorted.txt");
    if (file) {
        for (int i = 0; i < size && file.good(); i++) {
            file >> nlist[i];
        }
        file.close();
    } else return false;

    return true;
}

void ArrayUtility::displayNumbers(int nlist[], int size) {
    for (int i = 0; i < size; i++) {
        cout << nlist[i] << " ";
    }
}
```

ArrayUtility.cpp is continued on next page.

ArrayUtility.cpp – continued.

```
void ArrayUtility::selectionSort(int nlist[], int size) {
    int min_val, min_idx;

    for (int i = 0; i < size; i++) {
        min_idx = i;
        min_val = nlist[i];
        for (int j = i + 1; j < size; j++) {
            if (nlist[j] < min_val) {
                min_idx = j;
                min_val = nlist[j];
            }
        }
        if (min_val != nlist[i]) {
            nlist[min_idx] = nlist[i];
            nlist[i] = min_val;
        }
    }
}

bool ArrayUtility::writeNumbers(int nlist[], int size) {
    ofstream file;

    file.open("sorted.txt");
    if (file) {
        for (int i = 0; i < size; i++) {
            file << nlist[i] << " ";
        }
        file.close();
    } else return false;

    return true;
}

double ArrayUtility::calcAvg(int nlist[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += nlist[i];
    }
    return static_cast<double>(sum) / size;
}
```