THE UNIVERSITY OF HONG KONG

COMP1117C Computer Programming I

Assignment 5

Deadline: 5:00 pm, Dec 4, 2017

The main purpose of this assignment is to let you gain some experience of implementing and using C++ classes and objects. It is quite similar to Assignment 3, though you need to implement two more operations on arrays. They are the equality operation and the trace operation, and they are defined as follows.

1. Equality: array A is equal to array B when every pair of corresponding elements are equal, namely A[i][j] = B[i][j] for 0 <= i,j <= Size-1.

2. Trace: The trace of an array is the sum of the elements on the main diagonal of the array. That is A[0][0]+A[1][1]+ … +A[Size-1][Size-1].

For simplicity, you can assume the arrays have the same number of rows and columns, and only contain integer numbers.

You need to handle three files for this assignment, namely, array.h, array.cpp and main.cpp. We will show the content of array.h and main.cpp below. You need to complete array.cpp, which contains the implementation of the member functions in array.h. You also need to give the implementation of two (ordinary) functions, namely input() and output(), in main.cpp. For submission, you need to submit ONLY the files array.cpp and main.cpp.

Warning: When using dynamic arrays, make sure they are properly deleted when you don't need them anymore. Your marks will be deduced if you fail to do so. Not deleting dynamic arrays may cause an abnormally high memory usage, and may even crash your computer.

Follwing is the content of array.h.

```
================================================================

#ifndef ARRAY_H
#define ARRAY_H

class Array
{
public:
   // The constructor creates a dynamic array v[s][s], and set size
```

```
to s. Then it initializes all entries v[i][j] to 0.
   Array(int s);
   // The destructor deletes v.
   ~Array();
   // replace v by the transpose of v
   void transpose();
   // return the sum of the diagonal elements of A
   int trace();
   // return v[i][j], the element at row i, column j
   int get(int i, int j);
   // return the size of this array
   int get_size();
   // set v[i][j] to value
   void set(int i, int j, int value);

   // replace v by the sum of v and B.v
   void operator +=(Array &B);
   // replace v by the sum of v and B.v (i.e. v - B.v)
   void operator -=(Array &B);
   // replace v by the product of v and B.v
   void operator *=(Array &B);
   // return true if v[i][j] is equal to B.v[i][j] for all i, j in
[0,size-1]; otherwise return false
   bool operator ==(Array &B);

private:
   // For the dynamic array v. Find out how to create an dynamic 2D
array yourself.
   int **v;
   // size of the array
   int size;
};

#endif
```

==========

Following is the content of main.cpp. The main function asks the user to input Size and two arrays A and B, both having Size rows and Size columns. Then the user indicates an operation code, and the program performs the corresponding operation and outputs the result. The result of the operation (except for trace and equality) will replace the content of array A.

==========

```
#include <iostream>
```

```
#include "array.h"
using namespace std;

/*
You should implement the functions input and output
*/
void input(Array &C)
{
  // give your implementation of input() here.
}

void output(Array &C)
{
  // give your implementation of output() here.
}

int main(void)
{
   // input size
   int size;
   cout << "Please input array size:" << endl;
   cin >> size;

   // input A, B
   Array A(size), B(size);
   cout << "Please input " << size << "x" << size << " integer array
A:" << endl;
   input(A);
   cout << "Please input " << size << "x" << size << " integer array
B:" << endl;
   input(B);

   int op;
   bool stop = false;
   while (!stop)
   {
      cout << "Please input operation number (0. Stop; 1. Transpose;
2. Addition; 3. Subtraction; 4. Product; 5. Trace; 6. Equality):" <<
endl;
      cin >> op;
      switch (op)
      {
      case 0:
         // Stop
```

```
            stop = true;
            break;
        case 1:
            // Transpose
            A.transpose();
            output(A);
            break;
        case 2:
            // Addition
            A += B;
            output(A);
            break;
        case 3:
            // Subtraction
            A -= B;
            output(A);
            break;
        case 4:
            // Product
            A *= B;
            output(A);
            break;
        case 5:
            // Trace
            cout << A.trace() << endl;
            break;
        case 6:
            // Equality
            if (A == B)
                cout << "A is equal to B" << endl;
            else
                cout << "A is not equal to B" << endl;
            break;
        }
    }
    return 0;
}
```

==========

Please remember to implement the input() and output() functions in main.cpp. The input and output format for an array are both (take a 3x3 array for example):

<A[0][0]><space><A[0][1]><space><A[0][2]><newline>

<A[1][0]><space><A[1][1]><space><A[1][2]><newline>

<A[2][0]><space><A[2][1]><space><A[2][2]><newline>

Sample input/output (blue texts are input):

```
Please input array size:
3
Please input 3x3 integer array A:
1 2 3
4 5 6
7 8 9
Please input 3x3 integer array B:
1 2 3
4 5 6
7 8 9
Please input operation number (0. Stop; 1. Transpose; 2. Addition;
3. Subtraction; 4. Product; 5. Trace; 6. Equality):
1
1 4 7
2 5 8
3 6 9
Please input operation number (0. Stop; 1. Transpose; 2. Addition;
3. Subtraction; 4. Product; 5. Trace; 6. Equality):
1
1 2 3
4 5 6
7 8 9
Please input operation number (0. Stop; 1. Transpose; 2. Addition;
3. Subtraction; 4. Product; 5. Trace; 6. Equality):
2
2 4 6
8 10 12
14 16 18
Please input operation number (0. Stop; 1. Transpose; 2. Addition;
3. Subtraction; 4. Product; 5. Trace; 6. Equality):
3
1 2 3
4 5 6
7 8 9
Please input operation number (0. Stop; 1. Transpose; 2. Addition;
3. Subtraction; 4. Product; 5. Trace; 6. Equality):
4
30 36 42
66 81 96
```

```
102 126 150
Please input operation number (0. Stop; 1. Transpose; 2. Addition;
3. Subtraction; 4. Product; 5. Trace; 6. Equality):
5
261
Please input operation number (0. Stop; 1. Transpose; 2. Addition;
3. Subtraction; 4. Product; 5. Trace; 6. Equality):
6
A is not equal to B
Please input operation number (0. Stop; 1. Transpose; 2. Addition;
3. Subtraction; 4. Product; 5. Trace; 6. Equality):
0
```

Notes:

1. You don't have to check that the input parameters to your functions are valid. We will run your code only with valid inputs.

2. For simplicity, you can assume the arrays have the same number of rows and columns, and only contain integer numbers.

3. You are not advised to modify array.h (though you may modify it for debugging). In fact, you need not to submit it, because we will use the original array.h given in the template for compiling your program. Thus, before submission, make sure your array.cpp works well with the original array.h.

4. You are not advised to modify the main function in main.cpp. You can output debug information in the main function, but your main function will be overwritten when you evaluate your program on Moodle. Any changes of the main function will be ignored, and the main function will be replaced by the original one given. Before submission, make sure your program works well with the original main function.

Important Notes:

1. You must write your name and university number as part of the comments at the beginning of your program source code.
2. Your program must follow the formats of the sample input/output strictly.
3. Your program should be properly indented and documented.
4. You should only submit source array.cpp and main.cpp, not executables.
5. You must use the program names as suggested in the specification, i.e., array.cpp

and main.cpp.

6. Please make sure that your source code could be compiled in Code::Blocks environment with the settings mentioned in workshop) before submission.

7. Handin: Submit your programs electronically using the Moodle system under Assignments section, as done in Assignment 1. Late submission will NOT be accepted.