**Solution for Homework 6**

**ECE 309 Fall 2019**

**Due: October 9, 2019**

Upload an electronic copy of your answers to Moodle under HW6.

*This is a shared google document. This means (1) it may change to clarify content, and (2) other people can view your comments on this file. If you have questions, you are encouraged to comment directly on this document, but* ***do not add your answers here****. Make a copy into your private Google Drive and then edit the document.*

*DO NOT ADD ANSWERS TO THE SHARED DOC! THAT’S CONSIDERED CHEATING!*

# 1. C++ Templates

[50 points/5 points each] Consider the following code and answer the questions in the table below. Try to answer the questions without compiling the code, then check your answers and fix them with the help of the compiler.

#include <iostream>

template <typename T, int N=10>

class Array {

private:

T array[N+1];

public:

Array() {

for(int i=0; i<N+1; i++) array[i] = 0;

}

T& operator [] (int index) {

if (index>N || index < 0)

return array[N];

else

return array[index];

}

template <typename S>

Array<T,N>& operator= (S &other) {

for(int i=0; i<N+1; i++)

array[i] = other[i];

return \*this;

}

};

template <typename T, typename S> T operator+ (T &t, S &s)

{

T res;

for(int i=0; i<10; i++)

res[i] = t[i] + s[i];

return res;

}

template <typename T, int N> std::ostream& operator << (std::ostream& out, Array<T,N> &array)

{

out << "{ ";

for(int i=0; i<N; i++)

{

if (i>0) out << ", ";

out << array[i];

}

out << "}\n";

return out;

}

int main() {

// code snippet runs here

return 0;

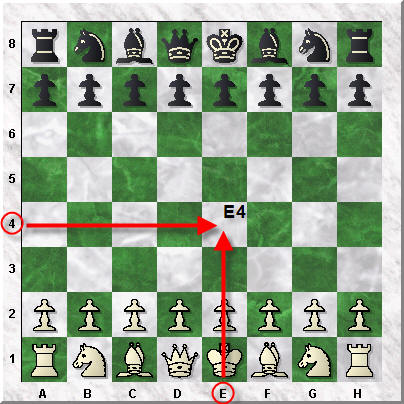
}

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Prompt** | **Snippet** | **Output / Error / Explanation** |
| 1 | What is the template specialization for variable x? | Array<int> x; | T=int |
| 2 | What is the output? | Array<int, 5> x;  std::cout << x; | {0,0,0,0,0} |
| 3 | What is the output? | Array<int> i;  Array<double> d;  i[1] = 10.5; d[1] = 5.5;  auto res = d+i;  std::cout << res; | { 0, 15.5, 0, 0, 0, 0, 0, 0, 0, 0} |
| 4 | What type is res deduced to be? (Hint: res must match the type on the right side of the assignment.) | Array<int> i;  Array<double> d;  i[1] = 10.5; d[1] = 5.5;  auto res = d+i;  std::cout << res; | Type is Array<double>. d+i has to use the operator+ and it will return a value of the same type as the left side. |
| 5 | What is the output? | Array<int> cvt;  char hello[10] =  "Hello!";  cvt = hello;  std::cout << cvt; | { 72, 101, 108, 108, 111, 33, 0, 0, 0, 0} |
| 6 | What is the specialization of operator= that is deduced by the compiler in this code snippet? | Array<int> cvt;  char hello[] =  "yoyoyoyo!";  cvt = hello; | T=int, N=10 (this is becaus cvt has T=int, N=10)  operator= is templatized on S, which should match type of hello. In this case, it’s a character array:  S=const char\* or char[] |
| 7 | What is the output? | Array<int> i;  for(int j=-1; j<100; j++)  {  i[j] = j;  }  std::cout << i; | { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9} |
| 8 | What is the output? Briefly explain how the code is calculating that result. | Array<int> i;  for(int j=-1; j<12; j++)  {  i[j] += j;  }  std::cout << i[10]; | Output is 20.  We only output i[10], which is the last position in the array.  Also, We only update that position when the index to operator[] is equal to 10, less than 0, or greater than 10.  That means, we accumulate the following cases:  i[-1] += -1  I[10] += 10  I[11] += 11  -1 + 10 + 11 => 20 |
| 9 | What’s wrong with this code, and how would you fix it without changing the class Array template? | Array<int> cvt;  char hello[10] =  "hello!";  cvt = hello + cvt; | Operator+ needs to return left side operand type. But, C/C++ does not permit returning array type. |
| 10 | What’s wrong with this code, and how would you fix it to compile without changing the class Array template? | Array<double> d;  d = d + d + d; | Operator+ expects references. But, the result of the first d+d produces a non-l-value, which does not have a memory location and cannot have its reference taken. So, this code is incompatible with operator+ as currently written. Either operator+ should have use refernces, or, we need to split it up into two different statements:  sum = d+d;  d = sum+d; |

# 2. ZyLabs

[50 points] Complete the following ZyLabs.

1. [25 points] ZyLab 8.23. Implement a chess-board style array access operator. When playing chess, the positions on the board are labeled using x,y coordinates, in which x is a letter from A-H and y is a value from 1 to 8, as shown here:

[](http://blog.chesshouse.com)

[http://blog.chesshouse.com](http://blog.chesshouse.com/)

The ChessBoard object overloads its array access operator to accept a string as its index, like this:

* + char& operator[](std::string index);
  + The string must take the form “xy” like “A1” or “H5” with no spaces. The function should return a reference to the corresponding game position. **For example, you may want to represent the board as a character array and return a position in the array.**  For simplicity, we will use a char type to encode the value of the game piece at a position, for example ‘K’ for king or ‘P’ for pawn.
  + If an illegal index is requested, return a reference to another location in memory outside of the board. In the provided code, you may use the err field.

1. [10 points] ZyLabs 14.4. Modify the DList from Lecture 5 to be a templatized class, wherein the type of the item is a template parameter. Your implementation should support DList<std::string> and DList<int>.
2. [15 points] ZyLabs 14.5. Template to find max value in an array. Your templatized code should work with any type of raw C array. It should take the array and the size of the array as arguments to the function.

template <typename A> A array\_max(A arr[], int length);

The template parameters should be specialized on the type of the array object.