ECE2036: Test #1
Fall 2014 – GTL

Name:_____

Directions:

- This is a closed book test; however, one double-sided sheet (letter-sized or A4) of HAND-WRITTEN notes is permitted.
- You may not use a calculator on this exam.
- Unless stated otherwise, justify your reasoning clearly to receive any partial credit.
- You must write your answer in the space provided on the exam paper itself. Only these answers will be graded. If space is needed for scratch work, use the backs of previous pages.

1. (*2 points)* For a C++ class object, the two access specifiers that we discussed in class are _____ and _____.

2. *(2 points)* When a member function is defined outside the class definition, the function header must include the class name and the _____, followed by the function name to "tie" the member function to the class definition.

3. (*2 points)* In C/C++ we can have pointers. A pointer has as its value a(n) _____.

4. *(2 points)* Name three different types of looping structures that can be used in C++.
   _____
   _____
   _____

5. *(2 points)* What is the difference between source code and object code?

6. (*10%)* Please add code as specified after the comments below. For the "pseudo-random" numbers that you generate you do not have to worry about setting the seed value.

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

//add function prototype for the global function specified below


int y = 100;

int main()
{
int y = 1;

//Add code here to add a random number
// between 1 and 25 to  the global variable y.
//Make this statement as short as possible!



//Call a function with a void return type that changes the
//local variable y by replacing its value with a randomly generated
// number between 0 and 100.



//Add a single line of code to print the value of the local y variable



return 0;

}

 //Show the implementation of the global function here that takes in a
//value by reference and replaces its value by a randomly
//generated number between 0 and 100.
//Make sure that this function has a void return type.
```

7. *(10%)* Please indicate the result that is printed from the following code

---

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

int main()
{
//EXAMPLE 1
int a =-2;
int b =8;
int c =1;
int f = 0;

f = a/c*b*b/c;

cout << "The result of example 1 is:" << f << endl;

//EXAMPLE 2
a = 3;
b = 9;
c = 4;
f = 0;

f = a+++b+++c;

cout << "The result of example 2 is:" << f << endl;


//EXAMPLE 3
a =2;
b =5;
c =5;
double real_a = 3.0;
double real_b = 5.0;
double real_c = 5.0;
double real_f = 0.0;


real_f = real_b/c+a/b*real_a;

cout << "The result of example 3 is:" << real_f << endl;


return 0;
}//end main
```

8. (10%) Please show the output of this program. Assume that the user enters 1 at the prompt.

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

int main()
{
 int value = 0;

cin >> value;

switch (value)
{
   case 0:
     cout << "yellow" << endl;
     break;
   case 1:
     cout << "red" << endl;
   case 3:
     cout << "blue" << endl;
   case 4:
     cout << "pink" << endl;
     break;
   default:
     cout << "We love METZ" << endl;
}//end switch

return 0;
}//end main
```

9. (10%) In the following program, you need to add ONE character of text in multiple locations to make the output of this program be a 3. Please indicate the change in the program below.

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

void function1( int );

int main( )
{

int number = 5 ;

function1 ( number ) ;

cout << number << endl;

return 0;
}

void function1 ( int num )
{
num -= 2;
}
```

10. (*10%)* Show the output that is produced by this C++ program.

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

void crazyIvan(double [] );

int main()
{

double dbArray[] = {1.6 , 2.71, 3.14};
double *ptr1 = NULL;
double *ptr2 = NULL;

ptr1 = dbArray;
ptr2 = &(dbArray[1]);

cout << fixed << setprecision(2);
cout << *ptr1 << " " << (int) (*ptr2) << endl;

ptr1 = ptr2 + 1;

cout << *ptr1 << " " << *ptr2 << " " <<  dbArray[2] << endl;

crazyIvan(dbArray);

cout << scientific << setprecision(1);
cout << *ptr1 << " " << *ptr2 << " " <<  dbArray[0] << endl;

}//end of main


void crazyIvan(double val[] )
{

double temp;

temp = val[1];
val[1] = val[2];
val[0] = temp;

}
```

11. (20%)

A. Please show an implementation of the setAge member function after the comments below. This implementation should compile *outside* the class interface.

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

class Professor
{  public:

    Professor();
    Professor(double,double,int);

    void setHeight(double);
    void setWeight (double);
    void setAge(int);
    void setAverageTeachingEffectiveness(double);

    double getHeight();
    double getWeight();
    int getAge();
    double getAverageTeachingEffectiveness();

  private:
     double height;
     double weight;
     int age;
     double averageTeachingEffectiveness;

}; //end of class interface
```

//--Show implementation of the setAge member function here

B. Please show the implementation of the constructor with the three arguments from the class interface on the previous page. Assume the first argument sets the initial value of the `height` data member. The second argument sets the initial value of the `weight` data member. The third argument sets the `age` data member. Also initialize the `averageTeachingEffectiveness` to zero in your constructor. This implementation should compile *outside* the class interface. You can assume that there are implementations for all set and get functions. Please use good programming practice to receive full credit.

C. Assume that your client has the following main function and would like to make an array of POINTERS to the Professor class objects. Please add lines of code as indicated below.

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

int main()
{

Professor * profPtrArray[10];


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

//Insert a line of code here to dynamically allocate a separate professor object
// for each pointer in the array.  I will start the line for you...
// You should use your  constructor that you created in part B
// to initialize the age to 40, weight to 190, and height to 6 for each professor
//object in this array.

profPtrArray[i] = _____;

}


//Please show a single line of code to print the averageTeachingEffectiveness of
//of the first element in the profPtrArray


_____;

}
```

12. (*10%)* What is the output of the following code?

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

void  function1(int &);
void  function2(int &);

int main()
{
  int value1 = 24;
  double value2 = 5.8;

  value1 = value2;

  function1(value1);

  cout << value1 << endl;

}

void function1(int &value)
{
     function2(value);
}

void function2(int &value)
{
     double temp = 19.6;

     temp = (value++ )+ temp;
     cout << temp << endl;

}
```

13. (*10%)* Please write a program that generates a text file that generates a random sequence of 100 prime numbers our of the set of prime numbers between 2 and 29. Call the output file `randomPrimes.txt`, and have each randomly generated prime number on its own line in the output file. How and why might you want to incorporate the srand() function into your code?

    Hint: Perhaps you could initialize an array with a list of prime numbers to help you on your way!

    int primes[] = {2,3,5,7,11,13,17,19,23,29};

ORDER OF PRECEDENCE CHART

Precedence	Operator	Description	Associativity
1	`::`	Scope resolution	Left-to-right

2	`++ --`
	`()`
	`[]`
	`.`
	`->`
		Suffix/postfix increment and decrement
		Function call
		Array subscripting
		Element selection by reference
		Element selection through pointer	

3	`++ --`
	`+ -`
	`! ~`
	`(`*type*`)`
	`*`
	`&`
	`sizeof`
	`new, new[]`
	`delete, delete[]`
		Prefix increment and decrement
		Unary plus and minus
		Logical NOT and bitwise NOT
		Type cast
		Indirection (dereference)
		Address-of
		Size-of
		Dynamic memory allocation
		Dynamic memory deallocation
			Right-to-left

4	`.* ->*`	Pointer to member	Left-to-right
5	`* / %`	Multiplication, division, and remainder	
6	`+ -`	Addition and subtraction	
7	`<< >>`	Bitwise left shift and right shift	
8	`< <=` `> >=`	For relational operators < and ≤ respectively For relational operators > and ≥ respectively	
9	`== !=`	For relational = and ≠ respectively	
10	`&`	Bitwise AND	
11	`^`	Bitwise XOR (exclusive or)	
12	`|`	Bitwise OR (inclusive or)	
13	`&&`	Logical AND	
14	`||`	Logical OR	

15	`?:`
	`=`
	`+= -=`
	`*= /= %=`
	`<<= >>=`
	`&= ^= |=`
		Ternary conditional[1]
		Direct assignment (provided by default for C++ classes)
		Assignment by sum and difference
		Assignment by product, quotient, and remainder
		Assignment by bitwise left shift and right shift
		Assignment by bitwise AND, XOR, and OR
			Right-to-left

16	`throw`	Throw operator (for exceptions)	
17	`,`	Comma	Left-to-right