Name: ______________________________________ Section: ___________________
Print legibly!

COMPUTER SCIENCE 261
PROGRAMMING CONCEPTS

EXAM 2 – VERSION 1
Fall 2014

150 Points

Absolutely ***no electronic devices*** may be used during this exam.

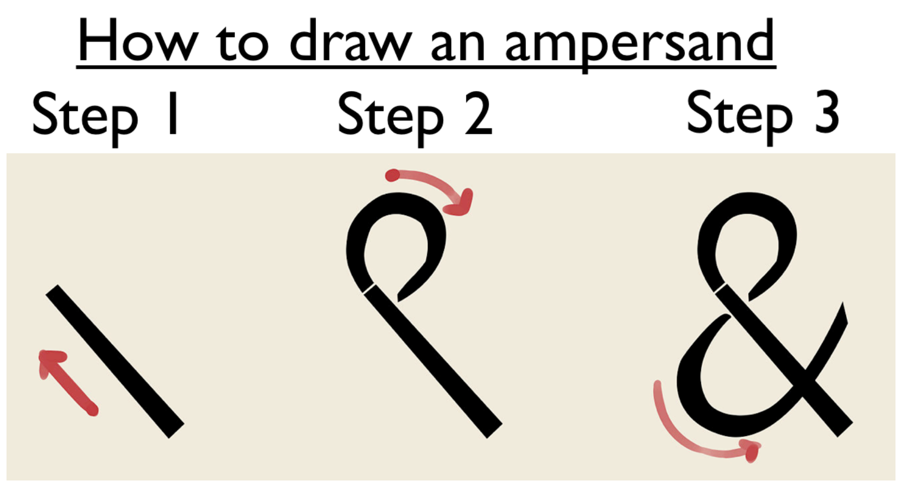
1. No cell phones, computers, calculators, music players, headphones, etc.
2. Set your cell phone to be ***completely silent***, not vibrate.
   If it vibrates during the exam, it will be confiscated.
3. Place these and any other electronic devices in your backpack, ***out of immediate reach***.

Failure to follow these instructions will result in a ***zero for the exam***.

You are expected to complete this exam ***in its entirety*** before leaving the exam room.

CLOSED BOOK – CLOSED NOTES

YOUR SCORE: _______ POINTS



All of your answers must be written in clear, readable text. ***<u>If we cannot read it, it is not correct</u>***.

1. 10 points. **True** or **False**. Write T or F to the left of each statement.

__**T**___ An `int` takes up more space than a `char` in memory.

__**F**___ Vectors can hold different types of elements inside the same object. In other words, I can have a vector that contains both `ints` and `doubles`.

__**T**___ Arrays are stored in contiguous blocks of memory.

__**F**___ It is NOT possible to create a multi-dimensional array that has more than three dimensions.

__**F**___ C-style strings are dynamic, i.e., the user can change their size during run-time.

__**T**___ To declare a 3x4 array of integers called `myArray` with all values set to 0 you can use:

```
int myArray[3][4] = { 0 };
```

__**F**___ Because of memory restrictions, we can only create multi-dimensional arrays of built-in data-types (`int`, `float`, `double`, `char`, and `bool`).

__**F**___ The default constructor is *always* called whenever a class instance (object) is created.

__**F**___ A global function in main.cpp has the same access to the members of a class as a member function of the class.

__**T**___ Setter functions always require a parameter.

2. 10 points. Consider the class declaration for a `Widget` below.
   a. How many constructors are declared in the class?
      FOUR
   b. How many default constructors are declared in the class?
      ONE
   c. List the names of the data members of the class.
      happy, fun, name
   d. How many member functions (both public and private) are declared in the class?
      TEN
   e. The member function `punch` is given a special name. What is it?
      - A private operator
      - The default constructor
      - **A helper function**
      - An accessor function

```
Class Widget
{
    public:
       Widget();
       Widget(double h);
       Widget(char n);
       Widget(double h, int f, char n);
       double getHappy();
       int getFun();
       void setFun(int f);
       void attack(Widget &w);
       bool havingFun();
    private:
       double happy;
       int fun;
       char name;
       int punch(Widget &w);
};
```

3. 5 points. Circle all the valid constructor prototypes for a class named `MyAccount`? Assume the appropriate header files and `namespace` statements are provided.
   a. `myAccount(double balance);`
   b. `MyAccount() const;`
   c. `void MyAccount();`
   d. `MyAccount(const string& accountNumber, double balance);`
   e. `MyAccount();`

4. 8 points. Draw a conceptual picture of the array that would be created by the following code. Include both the indices and values.

```
int data[] = { 42, 37, 23, 88, 51, 91, 12, 67 };
int x = 5;
data[x] = data[6];
data[6] = data[x];
data[x-1] = data[x] - 2;
data[1] = 1;
```

42	1	23	88	10	12	12	67
0	1	2	3	4	5	6	7

5. 10 points. Suppose your program contains the following class definition:

```
class Automobile
{
  public:
      Automobile();
      Automobile(double newPrice, double newProfit);
      void setPrice(double newPrice);
      void setProfit(double newProfit);
      double getPrice() const;
      double getProfit() const;
  private:
      double price,
      double profit;
};
```

and suppose the main part of your program contains the following declarations.

```
Automobile prius, jaguar;
double aPrice, aProfit;
```

First state YES or NO for each of the following statements on whether the statement would compile in the main part of your program. For those that you answer NO, briefly explain why not.

a). `aPrice = prius.getPrice;`
NO … missing ( )

b). `prius.price = 25999.99;`
NO … price is a private data member

c). `aProfit = jaguar.getProfit();`
YES

d). `jaguar.setPrice(37999.97);`
YES

e). `Automobile myAuto(429.5, 67.8);`
YES

6. 2 points. Which of the following statements are TRUE? Circle all that apply.
   a. Objects are required to be passed by reference.
   b. A header file is where data members of a class are defined.
   c. getters and setters provide managed access to data members.
   d. The class implementation file is where function prototypes are declared.

7. 2 points. Which of the following does a constructor perform? Circle all that apply.
   a. Construct a new class.
   b. Initialize prototypes.
   c. Construct a new function.
   d. Initialize objects.

8. 14 points. Given the following declarations:

```
const int NROWS = 3;
const int NCOLS = 4;
double numbers[NROWS][NCOLS];
double sums[NROWS];
```

In both questions below, write code with `NROWS/NCOLS` (so values above can easily be changed).

a. Provide code to set all values in the `numbers` array to 5.

```
for (int i = 0; i < NROWS; ++i)
   for (int j = 0; j < NCOLS; ++j)
       numbers[i][j] = 5;
```

b. Provide code to sum the amounts in each row of the `numbers` array and place the results in the corresponding position in the `sums` array.

```
double sumIt = 0;
for (int i = 0; i < NROWS; ++i)
   {
      for (int j = 0; j < NCOLS; ++j)
          sumIt += numbers[i][j];
      sums[i] = sumIt;
      sumIt = 0;
   }
```

9. 9 points. Fill in each of the 10 blanks.

   a. Consider the declaration below. The size of the array in memory is ___**FOUR**____ elements.
      ```
      int a[4] = {0,0};
      ```
   b. We use the keyword ____**const**_____ to ensure that a function cannot modify the contents of an array passed to it.
   c. To loop through all of the elements of a 5x8 2-D array you would use an index for the second dimension that goes from __**0**____ to ___**7**____.
   d. In a 2D array, the first and second dimensions are normally associated to the ___**ROWS**___ and __**COLUMNS**___ of a table, respectively.
   e. To pass an integer to a function by reference, we use ___**&**_____.
   f. To access member functions of the string class, we use the ____**DOT OPERATOR**___.
   g. Given that an integer takes 4 bytes of storage, and myNumbers[1] is stored at memory location 240, the address of myNumbers[3] is ____**248**____.
   h. In general, if a class has R private data members, you expect to have __**R**__ setter functions and __**R**__ getter functions.

10. 4 points.
    a. Write ONE statement that declares a one dimensional character array (called `arr`) with four elements, such that each of the elements are initialized to 'A'?

       char arr[4] = { ‘A’, ‘A’, ‘A’, ‘A’};

    b. Considering a character requires 1 byte of storage, how much memory is needed to store the array declared above?

       4 bytes

11. 6 points. Draw a conceptual picture of the following array for THREE passes of the selection sort algorithm, which was described in both class and in your text required reading. In this example, data should be sorted in **descending** order. Show each pass of the algorithm in the space provided.
    ```
    int data[] = { 25, 30, 101, 76, 5, 99, 15, 64 };
    ```

101	30	25	76	5	99	15	64

101	99	25	76	5	30	15	64

101	99	76	25	5	30	15	64

12. 14 points. Write a function implementation (called `diffMinMax`) that has a 1D integer array and array size as its parameters. The function returns the difference between the maximum and minimum numbers in the array. Your function should NOT use more than one `for` loop; that is, your function should find the max and min numbers in a single pass of the array. Also ensure that the contents of the array being passed **cannot** be changed.

```
int diffMinMax(const int a[], int size)
{
    int min = a[0];
    int max = a[0];
    for (int i = 1; i < size; ++i)
         {
              if (a[i] < min)
                 min = a[i];
              if (a[i] > max)
                 max = a[i];
          }
      return (max – min);
   }
```

13. 12 points. Given the following function header:

```
Foo Foo::doIt (const Foo& f) const
```

a. The FIRST Foo in the function header is the (be specific): ___**return type**____

b. The SECOND Foo in the function header is the (be specific): __**class name for member fnc**_

c. The THIRD Foo in the function head is the (be specific): ____**parameter type**___

d. What is the purpose of the FIRST const (be specific)?

to ensure the argument that is passed to the function does not change

e. What is the purpose of the SECOND const (be specific)?

to ensure the object that calls the function does not change

14. 8 points. Consider the C++ source below. It does not show some of the Time class' member function implementations, but you may assume these are implemented and "do the right thing." Write what will be printed for the values of the A and B calls to `getSecs()`.

3 -1
3 -3
0 -3
0 9

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

class Time
{
    public:
        Time( double seconds );
        void setSecs( double seconds );
        double getSecs() const;
        void doTheTimeWarp( const Time& warpFactor );
        void doTheTimeWarp( double warpFactor );
    private:
        double secs;
};

void Time::doTheTimeWarp( const Time& warpFactor )
{
    secs = secs * warpFactor.secs; // multiply
}
void Time::doTheTimeWarp( double warpFactor )
{
    secs = secs + warpFactor; // add
}

int main()
{
    Time A(3), B(-1);
    cout << A.getSecs() << " " << B.getSecs() << endl;
    B.doTheTimeWarp( A );
    cout << A.getSecs() << " " << B.getSecs() << endl;
    A.doTheTimeWarp( B.getSecs() );
    cout << A.getSecs() << " " << B.getSecs() << endl;
    B.doTheTimeWarp( B );
    cout << A.getSecs() << " " << B.getSecs() << endl;

    return 0;
}
```

15. 16 points. Write the declaration of a new class, named `Thermostat`. Use `const` whenever appropriate. Your `Thermostat` class must have:
    a. One private double data member that holds the temperature (in Fahrenheit) at a given hour.
    b. One private integer data member that holds the hour (0 to 23) that the temperature was taken.
    c. A default constructor.
    d. A parameterized constructor that sets both the temperature and the hour.
    e. Accessor member functions for each data member.
    f. A member function that returns the callee's temperature in Celsius (i.e., converts Fahrenheit to Celsius. Note: C = (F – 32)/1.8.

```
class Thermostat
{
   public:
      Thermostat();
      Thermostat( double t, int h );
       double getTemp() const;
       void setTemp(double t);
       int getHour() const;
       void setHour(int h);
       double convertToCelsius();
   private:
      double temp;
      int hour;
};
```

16. 5 points. Using your `Thermostat` class from problem 15, write one of the get functions.

```
double Thermostat::getTemp() const
{
      return temp;
}
```

17. 5 points. Using your `Thermostat` class from problem 15, write the implementation for the parameterized constructor.

```
Thermostat::Thermostat(double t, int h)
{
      temp = t;
      if ( (h >= 0) && (h < 23) )
         hour = h;
      else
         hour = 0;
}
```

18. 10 points. Using your `Thermostat` class from problem 15, write a main function that:
    a. Declares two `Thermostat` objects that are initialized with (72.1, 14) and (52.8, 4).
    b. Prints the temperature of the **first** (72.1, 14) `Thermostat` object in Fahrenheit.
    c. Sets the temperature of the **second** (52.8, 4) `Thermostat` object to 62.8.
    d. Prints the temperature of the object just modified in Celsius.

```
int main()
{
   Thermostat firstTemp(72.1, 14);
   Thermostat secondTemp(52.8, 4);
    cout << “The temp. of first temperature is ”;
    cout << firstTemp.getTemp() << “F” << endl;
    secondTemp.setTemp(62.8);
    cout << “The temp. of second temperature is now ”;
    cout << secondTemp.convertToCelsius() << “C” << endl;
}
```