Final Exam

STAY CALM

CSI 201: Computer Science I
Fall 2016

Professors: Shaun Ramsey and Kyle Wilson

DON'T PANIC

Question	Points	Score
1	18	
2	16	
3	12	
4	8	
5	10	
Total:	64	

Powers of 2 are super duper special!!

I understand that this exam is closed book and closed note and is to be completed without a calculator, phone, or other computer. I am **NOT** allowed to use any external resources to complete this exam. All of the work that I am submitting for this exam is my own, and has been completed in accordance with the Washington College Honor.

Name: ANSWER KEY Ph.D.

Signature: mmmm mm

Section: **10**: 11:30-12:20 **11**: 1:30 - 2:20

GENERALLY: PINK will be used for answers
BLUE for explanation
GREEN for additional code & explanation

1. Short coding questions

(a) [4 points] Write two loops: one `for` loop, and one `while` loop. Both of them should do the same thing: print the numbers 1 through 1000.

```
for (int i=0; i < 1000; ++i) {
    cout << i+1 << endl;
}
//i=1 + i <= 1000 is an option on both
// sides. then cout << i would be used
//there are MANY correct answers here
```

```
int i = 0;
while (i<1000) {
    cout << i+1 << endl;
}
```

(b) [2 points] Write a line of code that creates a new dynamic array of size `N` called `temperatures`.

```
static: double temperatures[N];
dynamic: double * temperatures = new double[N];   <- actual answer to this exam question
vector: vector<double> temperatures(N);   <- more likely question for this semester
```

(c) [2 points] Show how to release the memory used by the array in the previous question.

```
delete [] temperatures;//Not a very likely question, perhaps something related to
// how to use pointers w/ a class or a conceptual
// question related to "why" pointers.
```

(d) [2 points] Write a line of code that doubles the value of a variable called `x`.

```
x = x * 2;
```

(e) [2 points] Write a line or two of code that creates a variable called `Pi` and initializes it to 3.14159.

```
double Pi = 3.14159;
```

14 is NOT divisible by 3
10 is also NOT divisible by 3

Examples: 14 is divisible by 7
10 is divisible by 5
20 is divisible by 10

x divisible by y means that if we take x ÷ y we would get a whole number with no fraction or decimal left over

Modulus IS IMPORTANT and incredibly USEFUL!

(f) [2 points] Write a few lines of code that increase a variable called `x` by 2, only if `x` is an even number.

if(x%y==0) then x is divisible by y

```
if (x%2 == 0) {
    x = x+2;
}
```

// Modulus does a LOT of tricks for us in C.S. It can // tell us when a number is divisible by another [which we // are exploiting here for odd/even - divisble by 2 or not] We have also // used it to constrain the value of an index in situations where we are using a // circular buffer (an array that wraps around) like in the 2D Game of Life. // where we used (i+1)%N. And we also use it in rand calls to constrain the // values returned to be between 0 + some other number. Like in d6() we use rand()%6+1

(g) [2 points] In a few lines of code, define a function called `plusOne` that takes one parameter (a double called `x`) and returns `x+1`.

```
double plusOne(double x){
    return x+1;
}
```

This part gives us 0-5 then we +1 to get 1-6

(h) [2 points] Consider the following class definition:

```
class Coordinate {
public:
    double latitude;
    double longitude;
};
```

Show (in a few lines of code) how to a create new variable named `washcoll` of type `Coordinate` with latitude/longitude values of 39.22 and 76.06, respectively.

```
Coordinate washcoll;
washcoll.latitude = 39.22;
washcoll.longitude = 76.06;
```

2. Code Output.

(a) [10 points] Determine what is printed by each code snippet.

Code Snippet:	Output:

`int a = 4;`
`int b = 5;`
`cout << a * b << endl;`
`cout << a / b << endl;`
	20
	0

`int a = 17 - 4;`
`if (a % 2 == 0) {` → is false
`  a = 7;` → doesn't run
`}`
`cout << a;`
	13

`double x = 3.59;`
`if ((5.0 < x) && (x < 7.0))`
`  cout << x + 4 << endl;`
`else if ((5.0 < x) || (x < 7.0))`
`  cout << x + 3 << endl;`
`cout << x + 2 << endl;`
	6.59
	5.59

`int i = 0;`
`while (i <= 16) {`
`  i = i + 2;`
`}`
`cout << i << endl;`
i = 0, 2, 4, 6, 8, 10, 12, 14, 16, 18
16 is equal to 16
Nothing outputs until here
	18

`int data[5];` //static array
`for (int i = 0; i < 5; ++i)`
`  data[i] = i - 1;`
`cout << data[3];`
data[0] = -1
data[1] = 0
data[2] = 1
data[3] = 3-1 = 2
data[4] = 3
As a vector this line would be:
`vector<int> data(5);`
	2

When I solve this, I don't compute the whole vector on the left. Instead, I notice that only data[3] is output. Since the loop assigns to data[i], then data[3] is assigned when i=3. Thus data[3] gets the value of 3-1=2. This happens very quickly when thinking mentally & makes it so I don't have to compute the whole array or vector! left shown in green.

(b) [4 points] Consider this function definition:

these are by reference & thus "array" is a reference to the passed value!

```
void fillArray(double[] array, int N) {
    double temp = array[0];
    for (int i = 1; i < N; ++i)
        array[i] = temp;
}
```

vector<double> &array

make all other elements of the array/vector equal to the first element

What is the output when we run the following code snippet?

```
double * nums = new double[20];
nums[0] = 3.1;//or nums.at(0)=3.1;
nums[1] = 3.2;
fillArray(nums, 20);
cout << nums[0] << endl;
cout << nums[1] << endl;
```

//As a vector this would be: vector<double> nums(20);

//was changed in the function

3.1
3.1

(c) [2 points] Consider this function definition:

```
void f(int a, int b) {
    b = 2;
    a = a - b;
}
```

//notice this is call-by-value

What is the output when we run the following code snippet?

```
int a = 0;
int b = -1;
f(b, a);
cout << a << endl;
cout << b << endl;
```

since vars are call by value a/b unchanged in main

0
-1

You can imagine vector questions instead of array questions throughout.
Recursion is sparsely lacking on this exam! Make sure you can use them for output, understand the stack & can identify: infinite recursion, base cases, & recursive cases.

3. Concepts: answer each question briefly.

(a) 2 points What is an array index? What is a vector index? Which indices might be invalid?

A way to indicate an entry in a vector or array. The index describes which position an element in the array is requested. Indices begin with 0, so the first element in a vector is at index 0 & the last element in the vector is at the size of the vector/array minus one.

(b) 2 points What does `endl` do?

Puts a return (ends a line) for any output stream (consoles, files, or stringstreams)

(c) 2 points List the boolean operators.

&& and || (make sure you also know what !, ==, <=, >=, <, >, are called!)

(d) 2 points Why do we make class variables private?

to encapsulate & protect data. Allows use of classes without knowledge of underlying class data types

(e) 2 points Is this expression true, or false?

```
true && !(false || !false)
```

= true && !true = true && false

false

Step by step w/ "T" for true, "F" for false, "&" for and

T & !(F || !F) =
T & !(F || T) =
T & !(T) =
T & F =
F

//What's short-circuit Boolean evaluation & when does it happen?

(f) 2 points Which of the following is not a part of the process of turning C++ code into a working program? (Circle one)

- linking //Puts parts of a program together (ex: functions to prototypes)
- preprocessing //does all the #include statements (and other # statements)
- indexing ← not part of the process
- compiling //the actual translator. (c++ code to assembly code)

Other thoughts: How many times does this loop run?
What's a major situation where recursion might actually be used & beneficial?
Where's the major situation where pointers might actually be used?

4. [8 points] Debugging. The code below defines a function for asking the user to type in a positive number. It uses an input checking loop to keep asking until the answer is valid.

   Find all four bugs in this function. For each bug, write what the mistake was, and on what line it occurs (if possible). Each mistake is worth 2 points.

```
1.  int getPositiveNumber{
2.      while (x <= 0) {
3.          cout << "Type a positive number: ";
4.          cin >> x;
5.      }
6.      return x
7.  }
```

I will likely add this column to the final.

How/when does the bug occur? (logic, compile, run-time)

On this exam, all the bugs happen @ compile time. But, in Exam1, there were also some logical errors. Run-time errors exist for things like, integers ÷ 0, blowing vector bounds, dereferencing NULL, and so on.

Type: compile Error 1. No function parameter list (even if empty) Line #: 1

Type: compile Error 2.) should be " Line #: 3

Type: compile Error 3. No semi-colon Line #: 6

Type: compile Error 4. X is not declared Line #: 1-2

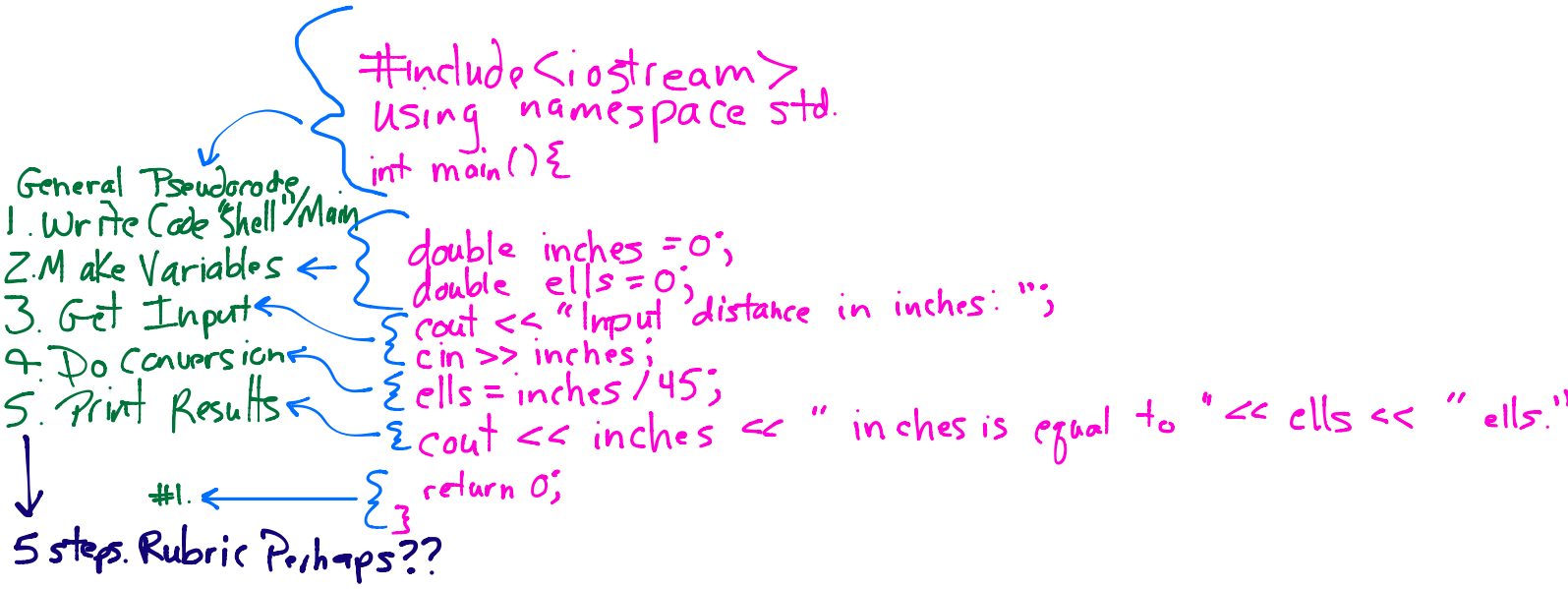
should be declared between these lines. Error occurs on line 2.

5. Programming.

   (a) [5 points] The *Ell* is an archaic unit of measurement. Write a unit converter program that converts from inches to ells. The conversion formula is:

```
ells = inches / 45
```

   The program should ask for an input distance in inches, and then it should print out the converted distance in ells. For this problem, you should write out every line of code that would go in your complete `.cpp` file.



(b) [5 points] Write a class definition for a class called `WaterBottle`. It should satisfy these specifications:

Hey looks 5 point question & there are 5 bullet points!

- It has one private variable: `current_amount`.
- It has a public function called `drink`, which takes a parameter `x` and reduces the `current_amount` by that much.
- It has a public function called `dump` which takes no parameters and sets `current_amount` to 0.
- It has a public accessor function called `getCurrentAmount`.
- It has a public one-parameter constructor which sets `current_amount` to a given value

```
class WaterBottle{
  private:
    double current_amount; //int might be okay too but types must match up!
  public:
    void drink(double x){ //students often have trouble with the concept that a
      current_amount -= x; //class member function can just modify the Object
      if (current_amount < 0){ //that called the function. Nevertheless, that's what
        current_amount = 0;     //many class member functions ultimately do
      }                         //this if statement is certainly
                                //not stated but we must
                                //infer this from the description
                                //given
    }
    void dump(){
      current_amount = 0;
    }
    double getCurrentAmount() const {
      return current_amount;
    }
    WaterBottle(double current_amount) {
      this->current_amount = current_amount;
      if (this->current_amount<0){ //this if requires even more insight into
        this->current_amount=0;    //the desired effect of this particular class
      }
    }
```