Exam 2

CSI 201: Computer Science 1
Fall 2016

Professors: Shaun Ramsey and Kyle Wilson

Question	Points	Score
1	18	
2	29	
3	18	
4	15	
Total:	80	

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 mine. I have completed this exam in accordance with the Washington College Honor Code.

KEY!

Name: _____

Signature: _____

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

Answers ARE ORANGE!

EXPLANATIONS ARE BLUE!

EXCLUSIONS AND OTHER COMMENTS ARE THIS COLOR!

1. **Concepts.** Answer each question briefly.

(a) [3 points] How many elements are in the following array?

```
bool on_time[43];
```

43

ARRAY QUESTION OUR EXAM IS W/VECTORS

```
vector<bool> on_time(43);
```

43

AS A VECTOR QUESTION

(b) [3 points] How are arrays and pointers related?

Array names are pointers.

THERE'S NO RELATED QUESTION FOR OUR EXAM2.

(c) [3 points] What important C++ rule does this function definition break?

```
int[] makeArrayOfTenZeros() {
    int arr[10];
    for (int i = 0; i < 10; ++i)
        arr[i] = 0;
    return arr;
}
```

NEVER RETURN A STATIC ARRAY ON OUR EXAM, THIS MIGHT BE A QUESTION REGARDING COPIES AND USING VECTORS IN RETURNS

this question as a vector question

(d) [3 points] Describe one possibility that could happen when this code is run:

```
double * four_elements = new double[4];
cout << four_elements[100];
```

It might segmentation fault or it might simply "work" to display an arbitrary value

what happens when the following code is run.

```
vector<double> four(4);
cout << four.at(4);
```

It crashes due to index out of bounds

(e) [6 points] Explain the differences between static and dynamic arrays.

	static	dynamic
size known at	compile time	run time
size is	a constant	a variable
memory is allocated	on the stack	on the heap
is declared by	[SIZE]	new type [SIZE]

2. **Short Coding Questions.** Write a line or two of code to answer each question.

(a) 3 points Show how to call a function called `printInstructions`. Here is this function's prototype: `void printInstructions();`

```
printInstructions();
```

Make sure to know the differences between prototypes, definitions and calls.

(b) 3 points Write a line of C++ code to print out the value of the first element of an array called `zebras`.

output to the console!

```
cout << zebras[0] << endl;
cout << zebras.at(0);
```

vector

this works w/ vectors also but it is better practice to use at

(c) 3 points Show how to free up the memory that is being used by the array `arr`, which was created like this:

```
double * arr = new double[15];
```

```
delete [] arr;
```

NO EQUIVALENT VECTOR VERSION OF THIS

(d) 3 points Write a prototype for a function called `printArray` that takes as input an array of type `double` and an integer length of the array. The function will not return anything.

Remember from handout this means as a parameter

```
void printArray(double *, int);
```

Vector of doubles version of this question:

```
void printVector(vector<double>);
```

No second parameter is required for vectors because vectors "KNOW" their size.

(e) 3 points Give a line of code to change the last element of an integer array to be equal to the second element of that array. The array is called `puppies` and the size variable is called `N`.

IN MANY OF THESE QUESTIONS REPLACE THE WORD ARRAY WITH VECTOR

```
puppies[N-1] = puppies[1];
```

this works for vectors also but at is better

vector version:

```
puppies.at(puppies.size()-1) = puppies.at(1);
```

Remember, the last valid index and thus the last element of a vector is located at size()-1. This is because the first index starts at 0. So if you have 5 elements, they would be accessed by indices 0,1,2,3,4. 0 is first. 4 is last!

there are 5 of these

(f) [4 points] Show how to declare a new array of `doubles` with 10 elements in two ways: (1) as a static array, and (2) as a dynamic array.

static: `double v[10];`
dynamic: `double *v = new double[10];`

vector: `vector<double> w(10);`

(g) [5 points] Give several lines of code that set every element of an array to `-1`. You should assume that the array is called `choices` and that the length of the array is stored in a variable named `N`.

```
for(int i=0; i<N; ++i){
  choices[i] = -1;
}
```

works for vectors but the following checks array bounds via the .at function.

```
for(unsigned i=0; i<choices.size(); ++i){
  choices.at(i) = -1;
}
```

(h) [5 points] Write a for loop to print every even-indexed element of an array to the console. You may assume that array has been declared and initialized elsewhere and is named `sodas`, and that the size of the array is stored in an `int` named `num_drinks`.

```
for(int i=0; i<num_drinks; i=i+2){
  cout << sodas[i] << endl;
}
```

works for vectors too but a .size vectors have & use .at (when checking for indices is important.

```
for(unsigned i=0; i<sodas.size(); i+=2){
  cout << sodas.at(i);
}
```

Print every even-indexed element of a vector to the console

3. **Code Output.**

(a) [3 points] What is the console output of the following code snippet?

```
for (int q = 10; q > 4; q -= 2) {
    cout << q - 2 << endl;
}
```

Notice that this does not change q because there is no assignment (=)

walk through the loop

q is 10
10 is > 4.
output q-2 & a new line → 8
q is q-2 = 8
8 is > 4
output q-2 = 6 & new line → 6
q is q-2 = 6
6 > 4
output 6-2 & new line → 4
6 is 4
4 is not > 4

Output:

8
6
4

(b) [3 points] What value does the function call `foo(12, 12, 12)` return?

```
double foo(int num1, int num2, int num3) {
    num1--;
    if (num1 < num2)
        num3 = num3 - 2;
    return num3 - num1;
}
```

num1 becomes 11
11 < 12? YES
num3 is 10
10 − 11 is −1

ANSWER: −1

(c) [3 points] Consider this definition for the function `h`:

```
int h(int b, int a) {
    return b;
}
```

Notice a is passed as the first parameter. BUT this function names the first parameter b.

FUNCTION SIMPLY RETURNS VALUE OF THE FIRST PARAMETER

What is the output of these lines of code?

```
a = 4;
b = 8;
cout << h(a, b);
```

Like calling h(4, 8)

Output:

4

(d) [4 points] Suppose that the function `f` is defined like this:

```
void f(int arr[], int size, int i){
    if (i < size)
        arr[i] = 7;
}
```

What is the console output of this code snippet?

```
int numbers[4];
numbers[2] = -5;
f(numbers, 4, 2);
cout << numbers[2] << endl;
```

If vector<int> is used instead of an array then the answer might change slightly.

Call-by-value vector answer	Call-by-reference vector answer
−5	7

Output:

7

(e) [5 points] What is the console output of the following code?

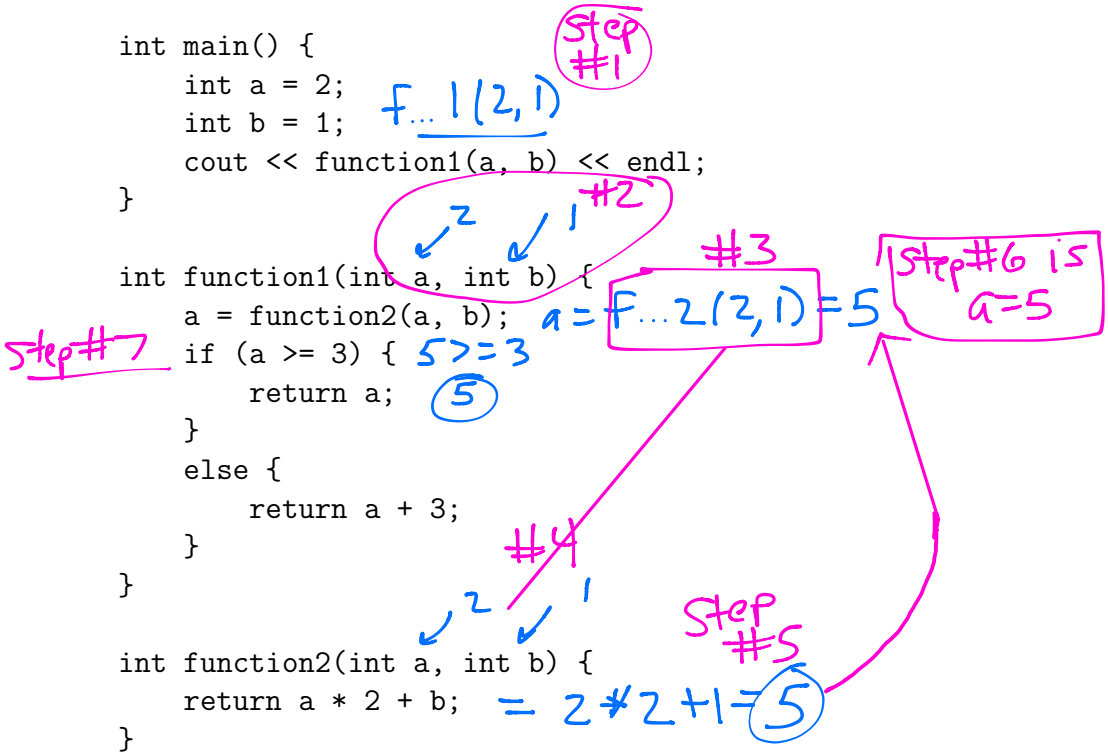
```
#include <iostream>
#include <string>
using namespace std;

int function1(int a, int b);
int function2(int a, int b);

int main() {
    int a = 2;
    int b = 1;
    cout << function1(a, b) << endl;
}

int function1(int a, int b) {
    a = function2(a, b);
    if (a >= 3) {
        return a;
    }
    else {
        return a + 3;
    }
}

int function2(int a, int b) {
    return a * 2 + b;
}
```



Output:

5

4. [15 points] Write a function to compute the evaluation of the algorithm described below. Use good programming practice, and choose proper return types and parameter values. Also write a `main` that demonstrates at least one function call of this function.

Algorithm: In math, the *length of a vector* of numbers is defined as the square root of the sum of the squares of the numbers. For example:

$$\begin{aligned}\text{length}([1,4,6,-2]) &= \sqrt{(1)^2+(4)^2+(6)^2+(-2)^2} \\ &= \sqrt{1+16+36+4} \\ &= \sqrt{57} \approx 7.55\end{aligned}$$

Write a function that computes this *vector length* for any array of numbers.

Clarification: You may *not* assume that input arrays always have 4 elements (like in the example). Your function must be general to any length array.

```
double length(const double *arr, const unsigned SIZE){
    double sum=0; //will hold sum of squares
    if(arr == NULL)
        return 0;
    for(unsigned i=0; i<SIZE; ++i){
        sum += arr[i] * arr[i];
    }
    return sqrt(sum);
}
```

AS A VECTOR QUESTION;

(WE HAVEN'T LEARNED THIS CONST) — to avoid a copy use call by reference

```
double length(const Vector<double> &arr){
    double sum=0;
    for(unsigned i=0; i<arr.size(); ++i){
        sum += arr.at(i)*arr.at(i);
    }
    return sqrt(sum);
}
```