Exam 2

CSI 201: Computer Science 1
Fall 2017
Professor: Shaun Ramsey, Ph.D.

It matters!

I understand that this exam is closed books and closed notes 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. The work that I am submitting and that I have viewed during this exam is mine. I understand that images, video and sound may be taken and recorded during this exam. I have completed this exam in accordance with the Washington College Honor Code.

For full credit, remember to use good style and programming practice throughout.

Key!

Name: _____

Signature: _____

CS1 SUPER FUN!

- Answers
- Additional Explanation
- Extra Commentary!
  - For example, there are some style typos denoted with "TAB →" indicating a tab indent should appear

1. 20 points Write the output that each code snippet would produce. Leave it blank if there is no output. Each line of ouput is worth 2 points. Assume that the code snippets are continuous. They are separated here to correlate output with your answers.

This is slightly different because the code is really one code snippet.

Code Snippet:	Console Output:

`//fill up the vector`
`vector<int> filler;`
`for (int i = 0; i < 11; i++) {` runs 11 times
TAB → `filler.push_back(i + 3);`
`}` so this must add 11 #s 0+3 through 10+3
`//output some of the entries`
`//and other information`
`cout << filler.at(0) << endl;` 0+3
`cout << filler.at(2) << endl;` 2+3
`cout << filler.size() << endl;` 11 pushed
	3
	5
	11

`//output some entries`
`//using a strange start` loop uses i=3,4,5,6
`for (int i = 3; i < 7; i++) {`
TAB → `cout << filler.at(i + 1) << endl;`
`}` so 4,5,6,7 for indexes
	7
	8
	9
	10

`//moving some values`
`//walk through this carefully`
lineA → `filler.at(2) = filler.at(3);` so (2) was 5 now (2) is 6
lineB → `filler.at(3) = filler.at(1);` (3) is now 4
lineC → `filler.at(1) = filler.at(2);` (1) is 6 b/c (2) is 6
`cout << filler.at(1) << endl;`
`cout << filler.at(2) << endl;`
`cout << filler.at(3) << endl;`
	6
	6
	4

0 – 3
1 – 4
2 – 5
3 – 6
4 – 7
5 – 8
6 – 9
7 – 10
8 – 11
9 – 12
10 – 13

2 pts each output

list is [3, 4, 5, 6, 7, ...]
after A [3, 4, 6, 6, 7, ...] // because .at(2) = .at(3) (=> 6)
after B [3, 4, 6, 4, 7, ...] // .at(3) = .at(1) (=> 4)
after C [3, 6, 6, 4, 7, ...] // .at(1) = .at(2) (=> 6) } .at(2) was 5, but "A" // changed the value to 6

.at(1) .at(2) .at(3)

2. Concepts: Answer the following briefly. When code is requested, your response should consist of less than 2-3 lines of code.

(a) 6 points What does fun1 return in the following function calls:

```
fun1(33.0);
```

true

```
fun1(0.0);
```

false

Although comparing doubles (floats) is a "sketchy" proposition that should be done in ranges.

if (fabs(funny) < 1e-6)

compares funny to see if it is "close" to 0.

```
fun1(-33.0);
```

true

```
//f1 is given by this function
bool fun1(double funny) {
    if (funny != 0) { //truly funny
        return true;
    }
    else { //not so funny
        return false;
    }
}
```

(b) 2 points Name one situation in which we use call by reference and explain why.

Input functions with more than one input to allow the inputs to be changed

(c) 2 points What happens when an invalid index is accessed in a vector?

An out of bounds error appears (crashing the program if using .at)
If using [] as the accessor then results are unpredictable/unknown.

(d) 6 points What does squareMax return when the following are called?

```
squareMax(4, 3);
```

3

```
squareMax(3,11);
```

9

```
squareMax(3, 9);
```

9

```
double squareMax(double var, double limits) {
   double value = var * var;
   if (value > limits) {
      value = limits;
   }
   return value;
}
```

TAB

(e) 4 points Examine the function squareMax from above. If we pass 9 as the second parameter, for what values of the first parameter will the function return 9? This is extremely useful, for example, if we wanted to write a unit test in a for loop to test a variety of values for the first parameter. In that situation, we might be able to assert that the function returns 9 in these situations. Give the full range of possible values for full credit. List a handful of values for partial credit.

(3,9) from (d) works,
(2,9) does not
(4,9) does. (5,9) does, (6,9), ...

For all values greater than or equal to 3 or less than or equal to -3

(f) 2 points Show a function call of the function squareMax above that returns a value of 13.

squareMax(10,13);

the choice here is any value big enough to give more than 13 when squared. So anything 4 and above is great. Really any value > 3.6 is fine but there's no reason to use this value.

(g) $\boxed{\text{4 points}}$ Demonstrate how to read in (from the console) a vector of 3000 strings named **names**.

```
for (int i = 0; i < 3000; ++i) {
    string s;
    cin >> s;
    names.push_back(s);
}
```

In this solution, names is assumed to be created with no size
Ex: vector<string> names;

```
for (unsigned i = 0; i < names.size(); ++i) {
    string s;
    cin >> s;
    names.at(i) = s;
}
```

In this solution names is assumed to be created with space reserve for the 3000 strings
Ex: vector<string> names(3000);

Always list your assumptions

(h) $\boxed{\text{4 points}}$ Demonstrate how to output (to the console) every other element of a vector of doubles named **grades**.

Change this 0 to a 1 to output the items in "odd" index positions instead of "even" ones

Skips every other. Other options involve using %

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

(i) $\boxed{\text{2 points}}$ Describe the run-time error with the following code snippet.

```
vector<int> pumpkins(42);
for(unsigned i = 1; i <= pumpkins.size(); i++) {
    pumpkins.at(i) = 42 - i;
}
```

← creates a vector with 42 entries

← loop starts w/ i=1 and runs until i=43, it enters the loop with i=42 but not with i=43

Error: when i=42, index is out of bounds.

3. $\boxed{\text{10 points}}$ In this problem, you have been given a vector of integers that represent ages and is aptly named **ages**. You are required to determine how many elements of the vector fall between the values of 17 and 22 (inclusive). Output (to the console) this amount as well as the total number of elements that are in the vector.

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

int main() {
    // assume the vector of integers named ages is defined here and
    // has been filled with all the data required
    ... ← ages is made & filled by someone else
    // Your code goes here
    int count = 0;
    for (int i=0; i< ages.size(); ++i) {
        if (ages.at(i) >= 17 && ages.at(i) <= 22) {
            count ++;
        }
    }
    cout << count << " out of " << ages.size()
        << " were between 17 and 22" << endl;

    // end of your code
    return 0;
}
```

~2 pts each

1) Make some count
2) For all elements
3) — check if b/n 17 & 22
4) ——— increase count
5) Output count & total #

4. 10 points Compound interest helps us understand how much money can be earned (or lost) by saving (or taking a loan). To compute how much money we will have (or owe) after a certain number of years (t) at a given rate (r) when we put in (or take out) a certain amount of dollars (P), uses the equation: $A = P(1+r)^t$. We've been asked to write a function in which P, r and t are given to the function. In return, the function computes and returns A. The cmath library provides the pow(x,y) function. pow(x,y) can be used to compute x^y.

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

double futureValue(double P, double r, double t);

int main() { //some unit tests
  assert( futureValue(1000, 0.0, 10) == 1000);
  assert( futureValue(1000, 0.1, 1) == 1100);
  assert( futureValue(1000, 1.0, 3) == 8000);
  assert( futureValue(1000, .01, 10) == 1104.52 );
    return 0;
}
double futureValue(double P, double r, double t) {
\\your code begins here
```

return P*pow(1+r, t);

Making intermediate variables would also be fine here & was more common in the answers!

Points:
a) Using pow correctly
b) proper return
c) proper calculation
d) proper usage of variables
e) style & function "pieces"

Roughly 2 pts each.

```
\\your code ends here
}
```

Question	Points	Score
1	20	
2	32	
3	10	
4	10	
Total:	72	