CSE2221 Midterm 1 Sample

1. Using the below method contract and header, what can we say about the result of powerTwo?

```
/**
* Calculate the square of a value within a relative error eps
*
* @requires i > 0 and eps > 0
* @ensures |powerTwo - i^2| / i^2 <= eps and powerTwo >= 0
*/
private static double powerTwo(double i, double eps){...}
```

What is the most accurate answer for the range of powerTwo(4, 0.5)?
A. [15.5, 16.5]
B. [8, 24]
C. [14, 18]
D. (15.5, 16.5)
E. None of the above

What about powerTwo(-2, 0.01)?
A. [3.98, 4.02]
B. [3.96, 4.04]
C. (3.96, 4.04)
D. exactly 4
E. None of the above

2. Fill in the following type table:

Expression	**Type**
xmlTree.child(5).child(1)	
(5 / 2) == 2.5	
256 + "1024"	
xmlTree.child(0).label()	
{1, 2, 3, 4, 5}	
4.0 / 2 * 7 / 10	

3. Consider the following string, what is the result of the 3 loops?

Loop #1:

```
string password = "Loganscarm3np4ssw0rd";

boolean hasUpper = false;
int i = 1;
while(i < password.length && !hasUpper){
     char c = password.charAt(i);
     if(Character.isUpperCase(c)){
          hasUpper = true;
     }
     i++;
}
```

i = _________
hasUpper = _________

Loop #2:

```
string password = "Loganscarm3np4ssw0rd";

boolean hasUpper = true;
for(int i = 0; i < password.length; i++){
     char c = password.charAt(i);
     hasUpper = Character.isUpperCase(c);
}
```

i = _________
hasUpper = _________

Loop #3:

```
string password = "Loganscarm3np4ssw0rd";

boolean hasLower = false;
int i = 0;
while(i < password.length() && !hasLower){
     char c = password.charAt(i);
     if(Character.isDigit(c)){
          hasLower = true;
     }
     i++;
}
```

i = _________
hasLower = _________

4. After calling the below method, what are the values of the variables?

```
private static double functionName(int x, int y){
     x = y;
     x -= 3;
     return y / x;
}

int j = 10;
int i = 2;
double x = functionName(i, j);
```

i = _________
j = _________
x = _________

5. We wish to perform max-pooling on an array of integers. Given an array of size 10 and filter size 3, the result will be an array of size [arraySize - filterSize + 1], in this case it will be [10 - 3 + 1] = 8.

Max-pooling is done as following: Given some sub-array that is equal to the filter size (in this case, 3), condense the 3 values into 1 value that is the maximum value in the sub-array.
For example:

```
int[] numbers = {1, 3, 4, 6, 1, 7, 8, 10, 11, 0}
```

The result of max-pooling is:

```
{4, 6, 6, 7, 8, 10, 11, 11}
```

This is because on the first iteration we have the sub-array [1, 3, 4], the maximum of this sub-array is 4. On the next iteration, we shift the filter up by 1 (this is called stride). Our new sub-array is [3, 4, 6], the maximum of this sub-array is 6. On the next, now 3rd, iteration, we shift our filter up by 1 again. Our new sub-array is [4, 6, 1], which has a maximum value of 6.

The iterations will continue until we reach the final sub-array, which is [10, 11, 0]. The maximum of this sub-array is 11. After this, we have succesfully completed max-pooling.

Given the function "max" and the "main" function below, implement the function "maxPool" below main to where it satisfies the problem mentioned above.

```
/**
* Determine the maximum value in an integer array.
*
* @requires arr.length > 0
* @ensures [the maximum value in arr is returned]
*/
private static int max(int[] arr){...}

public static void main(String[] args){
     int[] numbers = {7, 3, -8, 1, 5, 2, -3, -4, 10, 2, -11, 6, 3, 1, 0, 9};
     int filterSize = 3;
     numbers = maxPool(                                      ); // You choose what is passed to maxPool
}

// Implement maxPool as a public method below this line
```

This page was intentionally left blank for extra space for #5

6. Complete the method body for "containsMirror" so that it satisfies the method contract. Do **NOT** use multiple returns or a break statement. Your implementation should also stop early if a mirror is found.

A mirror is defined as the following: mirror(x) = -x. For example, +6 is the mirror of -6 and -42 is the mirror of +42.

```
/**
* Returns true if the array contains mirror values, false if not.
*
* @requires [arr does not contain 0]
* @ensures containsMirror = [arr contains mirror values]
*/
private static boolean containsMirror(int[] arr){









}
```

7. Draw the tree for the following XML document:

```
<cse2221 size="40">
    Software 1: Concepts
    <lecture time="4:10PM" room="DL369"/>
    <lab time="4:10PM" room="PO155"/>
    <instructor gta="true">Logan Frank</instructor>
    <students>
        <student year="2">Paolo</student>
        <student name="Bettina" year="1"></student>
    </students>
</cse2221>
```

8. Answer the following multiple choice questions

8.1. Is the sum of two irrational numbers irrational?
A. Yes
B. No
C. Sometimes

8.2. What is the implementer's responsibility?
A. Meeting the postconditions
B. Having efficient code
C. Writing clean code with comments
D. All of the above

8.3. Fill in the blank and answer the question: What is "num" in the _________ Math.abs(num)?
A. statement, argument
B. expression, argument
C. statement, formal parameter
D. expression, formal parameter

9. Tracing table, you know the drill.

Code	**State (Variable Names**

	str = “Ohio State”
	i = 9
	odds = “”
	evens = “”

while (i >= 0) {	
	str = i = odds = evens =
if (i % 2 == 1) {	
	str = i = odds = evens =
odds += str.charAt(i); {	
	str = i = odds = evens =
} else {	
	str = i = odds = evens =
evens += str.charAt(i);	
	str = i = odds = evens =
}	
	str = i = odds = evens =
i -= 1;	
	str = i = odds = evens =
}	

10. Recall from project 2 that Newton iteration can be used to compute the square root of a number with:

$$\text{repeatedly assign r = (r + x / r) / 2 until } (|r^2 \text{ - x}| \text{ / x}) < \epsilon^2$$

Now we wish to implement the additionally activity: modify root to perform the k-th root of x within relative error epsilon. Given the items below, derive the formula (so it is in the same form as what is used in the project, mentioned above) to use in Newton iteration to compute the k-th root of x within relative error epsilon.

$$\text{We want to find } r^k = x\text{, therefore we let } f(r) = r^k - x$$

Use the general form of the Newton-Raphson iterative equation (given below) to derive your solution, show your work (note f' is the derivative of f).

$$r_{n+1} = r_n - (f(r_n)/f'(r_n))$$

Now, implement the root function using your derived equation below. You may use Math.abs and Math.pow. Make sure no zero division is performed.

```
/**
* Computes estimate of k-th root of x to within relative error of epsilon.
*/
private static double root(double x, int k, double epsilon){






}
```

This page was intentionally left blank for scratch work.