Exam 2: Online Version

1. Which if statement below tests if letter holds R? (letter is a char variable)
   a. if (letter == "R") → This uses double quotes, so the system will interpret # as a literal String, not a char. This will cause an error.
   b. if (letter >= 'R') → This will not cause an error, but will return true for R,S,T,etc.
   c. if (letter == R) → This uses no quotes, so the system will interpret R as a variable. This will cause an error unless there is a char variable named R
   d. if (letter = 'R') → This uses a single =, which is an assignment operator instead of ==, which is a comparison operator. This will cause an error.
   e. if (letter == 'R') → This uses the correct operator and the single quotes

2. What are if statements used for in programs?
   a. Repeating commands
   b. Storing data
   c. Numeric calculations
   d. Numeric casts
   e. Making decisions

3. The following if statement tests the rainfall in New York's Central Park during the months of June, July and August.

```
if (low <= rain && rain <= high)
      System.out.println("Rainfall amount is normal.");
else
      System.out.println("Rainfall amount is abnormal.");
```

   It could be replaced with:

   I.
```
if (rain >= low) {
      if (rain <= high)
            System.out.println("Rainfall amount is normal.");
} else
      System.out.println("Rainfall amount is normal");
```

   II.
```
if (rain >= low) {
      if (rain <= high)
            System.out.println("Rainfall amount is normal.");
      else
            System.out.println("Rainfall amount is abnormal.");
} else
      System.out.println("Rainfall amount is abnormal.");
```

   III.
```
if (rain >= low)
      System.out.println("Rainfall amount is normal.");
else if (rain <= high)
      System.out.println("Rainfall amount is normal.");
else
      System.out.println("Rainfall amount is abnormal.");
```

   a. I only
   b. II only → This solution essentially takes the && in the if statement from the original statement and expands it into nested statements
   c. III only
   d. II or III
   e. I, II or III

4. What is output by the following code?

```
int x = 36 % 8;
if (x >= 10)
      System.out.println(1);
else if (x >= 8)
      System.out.println(2);
else if (x >= 6)
      System.out.println(3);
else if (x >= 4)
      System.out.println(4);
else
      System.out.println(5);
```

a. 1
b. 2
c. 3
d. 4 → 4 is greater than or equal to (namely equal to) 4
e. 5

5. Consider the code:

```
if (y == 0 || x*y > 10)
```

Which of the following is an example of short circuit evaluation?

a. if x*y > 10 is false it evaluates y == 0 → The second condition is never evaluated first
b. if x*y > 10 is false it doesn't evaluate == 0 → See A
c. if y == 0 is false it doesn't evaluate x*y > 10 → This is an Or statement, so both halves need to be false for the statement to be false. If the first half is false, then the second half needs to be evaluated.
d. if y == 0 is true it doesn't evaluate x*y > 10 → This is an OR statement, so if either half is true, the statement is true. If the first half is true, it doesn't need to evaluate the second half
e. if y == 0 is false it evaluates x*y > 10 → While this is true, it isn't an example of short circuit evaluation

6. The following truth table matches which boolean condition?

A	B	?
T	T	T
T	F	T
F	T	F
F	F	T

a. `A && ( A || B )` → This is equivalent to A which would not be true on line 2
b. `A || ( !A && !B )` → This is true whenever A is true OR both A and B are false as the table reflects
c. `A && ( A && B )` → This is equivalent to A && B which would not be true on lines 2 and 4
d. `!A && ( A || B )` → This is equivalent to !A && B, which would only be true on line 3
e. `A || ( A || B )` → This is equivalent to A || B, which would be true on line 3 and false on line 4

7. Consider the code:

```
if (a < b && c != d)
```

Which of the following is an example of short circuit evaluation?

a. if a < b is true it doesn't evaluate c != d → This is an AND statement – both halves need to be true for it to be true. If first half is true, the second half still needs to be evaluated
b. if a < b is false it doesn't evaluate c != d → This is an AND statement, so both halves need to be true for it to be true. If the first half is false, the statement is false and there is no need to evaluate the second half
c. if c != d is false it evaluates a < b → The second half will never be evaluated first
d. if c != d is true it doesn't evaluate a < b → See C
e. if a < b is true it evaluates c != d → While this is true, it is not an example of short circuit evaluation

8. `!( x < y && w == z)` is the same as which boolean expression?

a. `x <= y && w == z` → This is true when x<y and w==z, and the above statement is not
b. `x >= y || w != z` → This properly distributes the ! operator. It is true whenever the statement above inside the parenthesis is false
c. `z <= y || w != z` → This completely ignores the presence of x as a variable
d. `x <= y && w != z` → This is false when x<y and w !=z, and the above statement is true
e. `x < y && w != z`

9. Assume that x and y are boolean variables and have been properly initialized.

```
!(x || y) || (x || y)
```

The result of evaluating the expression above is best described as:

a. always true → The first half is true when both x and y are false, and the second half is true when either is true, therefore, because it is an OR statement, it is always true
b. always false → The statement is always true
c. true only when x is true and y is true → It is true when they are both true, but also under other circumstances
d. true only when x and y have the same value → It is true when they have the same value but also when they don't
e. true only when x and y have different values → It is true when they have different values but also when they don't

10. What is output to the screen by the following code?

```
int c = 2;
while (c < 6) {
      System.out.print( (int)Math.pow(-1, c) + " ");
      c++;
}
```

a. -1 1 -1 1 -1 1 -1 → The loop will only iterate four times
b. 1 -1 1 -1 → The loop will iterate four times, and the value of x at the first print statement will be 2, resulting in output of 1. At the end of the fourth iteration, the value of x will be 6, so there will not be a 5th iteration
c. -1 1 -1 1 -1 1 → See A
d. 1 1 1 1 1 1 → See A
e. -1 -1 -1 -1 -1 -1 → See A

11. How many times will the following loop repeat?

```
int num = 49;
while (num > 0) {
      if (num % 2 == 0)
            num++;
      else
            num--;
}
```

a. 20 b. 21 c. 22 d. 23 **e. Infinite Loop**

The loop will enter with num=49. 49% statement will be invoked. The loop will enter with num=49, etc.

12. What is output to the screen by the following code?

```
int num = 1987;
while (num > 0) {
      num = num/10;
      System.out.print(num%10 + " ");
}
```

a. 8 9 1 0
b. 198 19 1 0
c. 19 1 0 0
d. 7 8 9 1
e. The loop will not terminate

13. The following loop is intended to print the even numbers from 20 to 26:

```
int x = 20;
while (x < 26) {
      System.out.print(x);
      x++;
}
```

Which of the following changes would allow the code to work correctly?

a. The x++ needs to be x += 2 → This is true, but with this fix it would still fail to print 26
b. The x++ needs to be x += 2 and the x < 26 needs to be <= → This will print only the even numbers and allow 26 to be printed
c. The x < 26 needs to be <= → This is true, but the code would still print all numbers in the range, including the odd ones
d. It needs an if statement: if (x%2 == 0) → This could help in place of change x += 2, but the problem still exists with printing 26
e. Nothing, the code works as written. → It does not throw an error but it does not perform as intended

14. The following code is intended to input three integers and print the average:

```
System.out.println("Please enter three integers: ");
int a = scan.nextInt();
int b = scan.nextInt();
int c = scan.nextInt();
System.out.println("The average is: " + 1.0 * a + b + c / 3);
```

What is a potential problem with the code as written?

a. **It needs ( ) so the order of operations happens correctly. → It should be 1.0*(a+b+c)/3**
b. No correction needed, the code will work as written. → It will compile and run but will not produce intended output
c. It should be divided by 2, not 3. → There are three values so to take the average we divide by 3
d. It should use scan.nextDouble instead of scan.nextInt. → The input will be ints, so this is correct
e. The parentheses are not needed and will cause a mathematical error. → The parentheses are associated with the System.println call and are necessary. Their removal would cause an error

15. Which of the following needs a cast?

a. char stored in an int variable → int x='c' will store the ASCII code for c
b. **double stored in an int variable → int x=7.2 will throw an error, it needs to be cast because there is a loss of accuracy when storing a double in an int, so the system needs to be told that it's intentional. The other options don't need a cast because there is no loss in storing 7. As a double or 'c' as a String.**
c. char stored in a String variable → String x='c' will store "c"
d. int stored in a double variable → Double x=7 will store 7.0
e. char stored in a double variable → double x='c' will store the ASCII code for x as a double

16. Consider the following code segment:

```
int c = 1;
while (c <= 10) {
      if (c%3 == 1)
            System.out.print(c + " ");
      c++;
}
```

Which of the following produce the exact same output?

I.
```
int c = 1;
while (c <= 10) {
      c++;
      if (c%3 == 1)
            System.out.print(c + " ");
}
```

II.
```
int c = 1;
while (c <= 10) {
      System.out.print(c + " ");
      c += 3;
}
```

III.
```
int c = 0;
while (c <= 10) {
      c++;
      if (c%3 == 1)
            System.out.print(c + " ");
}
```

a. I only → Because this iterates before the if statement, it will fail to print the 1
b. II only → It's fine to increment by 3 instead of incrementing by 1 and checking a modular division
c. III only → This increments before the if statement but starts at 0, so it will still print the 1
d. II and III only → They are both correct
e. I, II and III → I is not correct

17. Which of the following correctly gives random numbers between -10 and 10 inclusive?

a. `int n = (int)(Math.random() * 20) - 10;`
b. `int n = (int)(Math.random() * 21) - 10;` → This is correct, the inclusive range (-10,10) has 21 total numbers
c. `int n = (int)(Math.random() * 11) - 20;`
d. `int n = (int)(Math.random() * 10) - 20;`
e. `int n = (int)(Math.random() * 10) - 21;`

18. Consider the following code:

```
int count = 4;
while (count <= 7) {
      count++;
      System.out.print(count + " ");
}
```

What are the first and last numbers output?

a. 4 7 b. 4 8 c. 5 7 **d. 5 8** e. Nothing is output.

C increments before the loop, so it will first print 5 and last print 8

19. Consider the following code:

```
int diff = 0;
if (Math.abs(num1 - num2) == (num1 - num2))
      diff = num1 - num2;
else if (Math.abs(num2 - num1) == (num2 - num1))
      diff = num2 - num1;
```

Which of the following will have the exact same result?

I.
```
int diff = Math.abs( num1 ) - num2;
```
II.
```
int diff = Math.abs( num1 - num2 );
```
III.
```
int diff = Math.abs( num2 - num1 );
```

a. I only → This takes the absolute value of num1 and then subtracts num2. It could be negative
b. II only → This is the absolute value of the difference of the two numbers
c. III only → This is the absolute value of the difference of the two numbers, the code above uses if statements to determine which direction the subtraction needs to be for the result to be positive, but that's superfluous
d. II and III only → These are both correct
e. I, II, and III → I is not correct

20. Of the following if statements, which one correctly executes exactly two commands only when the condition is true?

I.
```
if (y == 99) {
   System.out.println("A");
   System.out.println("B");
}
System.out.println("C");
```

II.
```
if (y == 99)
   System.out.println("A");
   System.out.println("B");
System.out.println("C");
```

III.
```
if (y == 99) {
   System.out.println("A");
   System.out.println("B");
}
```

a. I only
b. II only
c. III only → This will print A and B if y=99, or nothing elsewise
d. II and III but not I
e. I and III but not II

