CSC 102 Character WS

1. **Declare and assign a char variable called `myLetter` to z.**

```
char myLetter = 'z';
```

2. **Write a Java statement that makes `myLetter` uppercase.**

```
public static void main(String[] args) {
    char myLetter = 'a';
    myLetter = Character.toUpperCase(myLetter);
    System.out.println(myLetter);
}
```

3. **Write a java statement that finds out if `myLetter` is a digit.**

```
public static void main(String[] args) {
    char myLetter = '5';
    if (Character.isDigit(myLetter)) {
        System.out.println("myLetter is a digit");
    } else {
        System.out.println("myLetter is not a digit");
    }
}
```

4. **Write a java statement that finds out if `myLetter` is an upper case letter.**

```
public static void main(String[] args) {
    char myLetter = 'A';
    if (myLetter >= 'A' && myLetter <= 'Z') {
        System.out.println("myLetter is an upper case letter");
    } else {
        System.out.println("myLetter is not an upper case letter");
    }
}
```

5. **What is the output for `System.out.print(myLetter++);`**

```
public static void main(String[] args) {
    char myLetter = 'A';
    System.out.print(myLetter++);
}

Output: A
```

6. **What is the output for `System.out.print(++myLetter);`**
(do not include or base the result by what happened in #5)

```
public static void main(String[] args) {
    char myLetter = 'A';
    System.out.print(++myLetter);
}
```

Output: B

7. **What is the output for `System.out.print(myLetter--);`**
(do not include or base the result by what happened in #5 or #6)

```
public static void main(String[] args) {
    char myLetter = 'A';
    System.out.print(myLetter--);
}
```

Output: A

8. **What happens when a floating-point data type, like double, is cast into a `char`? (page 127)**
When you cast a floating-point data type, like double, into a char, something interesting happens. The conversion is a bit tricky, so let's break it down. First, the double value is converted into an integer by truncating the decimal part. It's like chopping off the extra bits after the decimal point. Next, this truncated integer value is further narrowed down to fit into a char. It's like squeezing a big number into a smaller container. The higher-order bits are discarded, leaving only the relevant bits. But here's the catch: when you cast a double into a char, you might lose some precision. Some details might get cropped out, and you end up with an incomplete picture. The resulting char value will represent a Unicode code point corresponding to the truncated integer, but it might not always give you the expected outcome due to the loss of precision.

9. **Evaluate the following Java statement: `char c = 90;`**
The Java statement char c = 90; is valid and will compile without any errors. In Java, a char is a 16-bit unsigned integer that represents Unicode characters. It can hold integer values ranging from 0 to 65,535 (inclusive). When you assign the value 90 to a char, the compiler interprets it as the Unicode code point and maps it to the corresponding character. In this case, the Unicode code point 90 represents the character 'Z'. Therefore, after the statement executes, the variable c will store the character 'Z'.

10. **What Java method in the Character class lets me know if my character is a letter?**
The Java method in the Character class that can be used to determine if a character is a letter is isLetter().