Name:_____ UNM Username:_____

Answer all questions in the space provided. Write clearly and legibly, you will not get credit for illegible or incomprehensible answers. Print your name at the top of every page. This is a closed book exam. Each student is allowed to bring one page of notes to the exam and is permitted the use of a "dumb" calculator to perform basic arithmetic.

Question:	1	2	3	4	5	6	7	Total
Points:	14	30	10	15	10	10	10	99
Score:								

The rest of this page is available as scratch space.

If you need to use additional paper, please write your name at the top of the page(s) and submit along with the exam.

1. Why don't the following code snippets compile? (Or do they?) Select the single correct answer for each from the following options and answer in the blank provided.

   A. Cannot compare int and short variables.
   B. Return value does not match method return type.
   C. Cannot use a keyword as a variable name.
   D. Cannot access private variable n from a public method.
   E. Cannot access n without an instance of MyClass.
   F. Variable n is out of scope in the main method.
   G. Variable n is a constant, so cannot be incremented.
   H. Did not initialize value of n.
   I. Invalid parameter type.
   J. Invalid return type.
   K. Invalid operator.
   L. Some other error.
   M. This code will successfully compile.

(a)
```
int single = 1;
int double = 2;
```
(a) _____ (2)

(b)
```
public class MyClass {
    private static final int n = 10;

    public static void main(String[] args) {
        ++n;
    }
}
```
(b) _____ (2)

(c)
```
public static void foo(int m, short n) {
    if(m = n) {
        System.out.println("Equal!");
    }
}
```
(c) _____ (2)

(d)
```
public static long bar(int n) {
    return (n % 2 == 0) ? n : 1;
}
```
(d) _____ (2)

(e)
```
public static boolean baz(int n) {
    return n;
}
```
(e) _____ (2)

(f)
```
int n = 5
```
(f) _____ (2)

(g)
```
public class MyClass {
    private short n = 5;

    public static void main(String[] args) {
        System.out.println(n);
    }
}
```
(g) _____ (2)

2. Write the answer in the blank provided.

(a) Which primitive data type could I use to store the value of the square root of 2? (2)

(a) _____

(b) Which primitive data type could I use to store the Greek letter π? (2)

(b) _____

(c) What is the value of the following expression? `1 + 2 * 3 + "4"+ 5 + 6` (2)

(c) _____

(d) What is the keyword used to make a variable that cannot be reassigned? (2)

(d) _____

(e) What is the keyword used to make a variable or method visible only to the class containing it? (2)

(e) _____

(f) What is the name for the special type of method that creates instances of a class? (2)

(f) _____

(g) The last index of an array is 12. How many elements can the array hold? (2)

(g) _____

(h) If I declare a `short` variable but do not initialize it, what is its value? (2)

(h) _____

(i) If I declare an `int[]` variable but do not initialize it, what is its value? (2)

(i) _____

(j) What is the keyword used to make a variable or method belong to a *class* rather than an *instance*? (2)

(j) _____

(k) What is the keyword used to access a member variable hidden by a local variable with the same name? (2)

(k) _____

(l) Name one of the types of branching statements in Java. (2)

(l) _____

(m) What is happening when a class contains two methods with the same name but different parameter lists? (2)

(m) _____

(n) What is is called when a method calls itself? (2)

(n) _____

(o) What keyword is used to indicate that a method does not return a value? (2)

(o) _____

3. The following Java program compiles and runs. What is its output? (10)

   Reminder: `System.out.print` will print its argument without going to a new line afterwards.

```
public class ArrayTest {

    public static final int ROWS = 2;
    public static final int COLS = 4;

    public static void fillArray(int[][] arr) {
        int x = 1;
        for(int c = COLS-1; c >= 0; c--) {
            for(int r = 0; r < ROWS; r++) {
                arr[r][c] = x;
                ++x;
            }
        }
    }

    public static void printArray(int[][] arr) {
        for(int r = 0; r < ROWS; r++) {
            System.out.print("[ "); // bracket at row start
            for(int c = 0; c < COLS; c++) {
                System.out.print(arr[r][c] + " "); // value and space
            }
            System.out.println("]"); // bracket and newline at row end
        }
    }

    public static void main(String[] args) {
        int[][] grid = new int[ROWS][COLS];
        fillArray(grid);
        printArray(grid);
    }
}
```

4. The following Java program compiles and runs. What is its output? (15)

```
public class MethodTest {

    public static String foo(int a, String[] strs) {

        int b = strs[a].length() % strs.length;
        String c = strs[b];
        System.out.println("a = " + a + ", b = " + b + ", c = " + c);

        if(a > b) {
            strs[a] = "final";
        } else {
            strs[b] = "exam";
        }
        return strs[a] + ", " + strs[b];
    }

    public static void main(String[] args) {
        int a = 3;
        int b = 1;
        String[] c = new String[]{"yay", "summer", "break", "time"};

        System.out.println(a + ", " + foo(a, c));
        System.out.println(b + ", " + foo(b, c));

        for(int i = 0; i < c.length; i++) {
            System.out.println(i + ", " + c[i]);
        }
    }
}
```

5. The following Java program compiles and runs. What is its output? (10)

```
public class TestObj {

    private int x;

    public TestObj(int x) {
        this.x = x;
    }

    public void printAndChange(int x) {
        System.out.println(this.x);
        System.out.println(x);
        this.x += x;
    }

    public static void main(String[] args) {

        int x = 2;
        TestObj first = new TestObj(x);
        first.printAndChange(x * 2);
        first.printAndChange(x + 3);
        --x;
        TestObj second = new TestObj(x);
        x += 7;
        second.printAndChange(x);
        x /= 3;
        second.printAndChange(x);
        x++;
        first.printAndChange(x);
    }
}
```

6. Write a method that takes an array of Strings and returns the length of the longest string in the array that does not contain the character "x", returning -1 if the array does not contain any strings without an "x" in them. You may assume there will not be any null values in the array. (10)

   To do this, copy exactly one line of code from each of the pairs below, place them in the correct order, and add indentation and closing curly braces as needed to make a correct Java method with the desired behavior.

   A. `public static int longestLengthNoX(String[] strs) {`

   B. `public static int longestLengthNoX(String strs) {`

   C. `int len = -1;`

   D. `int len = strs[i].length();`

   E. `int longest = -1;`

   F. `int longest = strs.length;`

   G. `if(len < longest) {`

   H. `if(len > longest) {`

   I. `if(!strs[i].contains("x")) {`

   J. `if(strs[i].contains("x")) {`

   K. `for(int i = 0; i < strs.length; i++) {`

   L. `for(int i = 1; i < strs.length; i++) {`

   M. `return len;`

   N. `return longest;`

   O. `longest = len;`

   P. `len = longest;`

7. Say that a "clump" in an array is a series of 2 or more adjacent elements of the same value. Fill in the blanks in the code to write a method that takes and array of integers and returns the number of clumps in it. (10)

   The following examples are to give you a feel for how it works.

```
countClumps( new int[] {1, 2, 2, 3, 4, 4} ) → 2
countClumps( new int[] {1, 1, 2, 1, 1} ) → 2
countClumps( new int[] {1, 1, 1, 1, 1} ) → 1
countClumps( new int[] {} ) → 0
countClumps( new int[] {1, 2, 3} ) → 0
```

   Use one option per blank. You will not use all of the options given below. You may just put the letter in the blank if that is easier than copying the code snippets.

A. `public static int countClumps(int nums)`	I. `int i = 0;`
B. `public static int countClumps(int[] nums)`	J. `int i = 1;`
C. `!inClump && nums[i-1] == nums[i]`	K. `int clumps = 0;`
D. `!inClump || nums[i-1] == nums[i]`	L. `int clumps = nums.length;`
E. `inClump = true;`	M. `return clumps;`
F. `inClump = false;`	N. `return i;`
G. `nums[i-1] != nums[i]`	O. `return nums;`
H. `nums[i+1] != nums[i]`	P. `return inClump;`

```
_____ {

    _____
    boolean inClump = false;

    for (_____ i < nums.length; i++) {

        if (_____) {
            clumps++;

            _____

        } else if (_____){

            _____
        }
    }

    _____
}
```