

This is a closed notes, closed book practice midterm.
... Actually you can do what you want, it's for your practice. I just recommend that.

Note: If you are asked to write code, you must declare all variables. If you are asked to just write a portion of code, you only need to write what is asked for. If you are asked to write a whole program, you must include everything (like the class, main, and import). Assume that whenever Scanner is used in the code given to you that this is at the top of the program: `import java.util.*;`

1. Given the following javadoc specification, show how to call and get a return value from the function `pow()`. Let `a` equal 2, `b` equal 7, and store the result in `c`. Print out the value that is returned. Assume `pow()` works.

```
static double pow (double a, double b)
Returns the value of the first argument raised to the power of
the second argument.
```

2. Write a method called `circleCircumference()` that takes as a parameter the radius of a circle of type `double` and returns its area. Avoid losing precision.

3. Given the following numbered lines of code:

```
1 class Foo {
2
3     public static void main(String[] args) {
4         int a = 5;
5
6         for(int b = 0; b < 100; b++) {
7             b = a*b;
8         }
9
10        int c;
11
12        c = foo(a);
13
14        System.out.println(c);
15    }
16
17    public static double foo(int d) {
18        double e = 1/d;
19        return e;
20    }
21 }
```

Write the line numbers of the lines that constitute the scope of each variable:

a: b: c: d: e:

4. What does this program print out? Why?

--

This is a closed notes, closed book practice midterm.
... Actually you can do what you want, it's for your practice. I just recommend that.

```
class TestProgram{
    public static void main(String[] args){
        int a = 4, b = 3;
        System.out.println(a);
        System.out.println(b);
        swap(a, b);
        System.out.println(a);
        System.out.println(b);
    }

    static void swap(int x, int y) {
        int temp;
        System.out.println(x);
        System.out.println(y);
        temp = x;
        x = y;
        y = temp;
        System.out.println(x);
        System.out.println(y);
    }
}
```

5. Arrays

a) Declare and create storage for an array of n integers called *Question5a*

b) Write a method called *Question5b()* that takes an array of integers and squares each element.

c) Show how you would call *Question5b()* with the array *Question5a* as a parameter.

d) After the call to *Question5b()*, is the original array changed in any way? Why? Try drawing a diagram to picture what the arrays and variables look like in memory.

6. Write a method called **arrayMax()** that takes an array of doubles as a parameter and returns the index of the largest element of the array.

This is a closed notes, closed book practice midterm.
... Actually you can do what you want, it's for your practice. I just recommend that.

7. Conway's Life program simulates cell life. It is "played" on a 2D array of elements that represent cells. It has three basic rules:
Rule 1. If a cell is dead and it has exactly three neighbors that are alive, it comes to life in the next generation.
Rule 2. If a cell is alive and it has fewer than 2 or more than 3 neighbors that are alive, it dies in the next generation.
Rule 3. Otherwise, the cell will be the same in the next generation as it is in the current generation.

Suppose that we have a life game board that is a 2D array of booleans, where the boolean value **false** means that a cell is dead and **true** means that a cell is alive. Write a method called **live()** that takes the board as a parameter and returns an updated board that shows what it looks like after exactly one generation. Try writing this method without worrying about the edges as `indexOutOfBounds` errors.

Additional Challenge: Assume that all edges are dead; they do not count as a neighbor. As extra practice, see if you can catch all the edges.

Recursion and Arrays Using For Loops:

Questions 8-11 – We will be working these out on the board during the review session. Use a separate sheet of paper to complete your answers, and compare them to what your classmates get.

For each question, write out the solution both iteratively (without recursion) and recursively (function calling itself).

Hint: Try breaking down each program into various parts, flowcharting it out, and then writing the code. I have written solutions to each and compiled them, so feel free to ask me for hints or solutions.

8. Write a class Fibonacci that takes in a number, and prints out onto the screen all numbers from 1 to the nth term of the Fibonacci sequence.

$f(0) = 1; f(1) = 1; f(i) = f(i-1) + f(i-2)$, for all $i \geq 2$

9. Write a class Powers that takes in a base of type int and power of type int, and computes the base raised to the power, and prints that to the screen.

Example: if base = 2 and power = 3, the program would output 8

10. Write a class RecursionABC that prints out onto the screen a sequence of char (not a String) from 'a' to 'z' on one line.

Hint: Remember, you can move from 'a' to 'b' by adding one to the char variable.

11. Write a class Palindrome that takes in a String, translates it into an array of char, then checks if the array is a palindrome. A palindrome is a sequence of characters that is spelled the same way forwards and backwards. For this assignment, treat the String as one word.

Examples: racecar level radar redivider
solutomaattimittamotulos (Finnish for the result from a measurement laboratory for tomatoes)

12. What does the following program print?

```
class Question12 {
    public static void main(String[] args) {
        int x = 2, y = 5;

        System.out.println("fun = " + fun(x, y));
        System.out.println("fun = " + fun(x+2, y));
        // be careful on the next one - a trick?
        System.out.println("fun = " + fun(y, x));
    }
    static int fun(int x, int y)
    {
        System.out.println("x = " + x +
                           ", y = " + y);
        return x - y;
    }
}
```

This is a closed notes, closed book practice midterm.
... Actually you can do what you want, it's for your practice. I just recommend that.

13. What is the output of the following program? What does the program do?

```
class Question13 {
    public static void main(String[] args) {
        int j = 2, k = 14;
        System.out.println("Now j and k are: "
            + j + " and " + k + ", and the function is " + what(j,k));
    }
    static int what(int x, int y) {
        System.out.println("comparing " + x + " and " + y);
        if (x < y)
            return x;
        else
            return y;
    }
}
```

14. What is the output of the following program? What does the program do? **Caution: This may be a trick question.**

```
class Question14 {
    public static void main(String[] args) {
        int x = 7, y = 14;
        System.out.println("Now x and y are: " + y + " and " + x +
            ", and the function is " + what(y,x)); // CAREFUL!!
    }
    static int what(int x, int y) {
        System.out.println("comparing " + x + " and " + y);
        if (x < y)
            return y;
        else
            return x;
    }
}
```

15. What does the following program print? Caution: a trick? What would the program print if we changed line 4 to `someNumber = timesTwo(someNumber);`

```
class Question15 {
    public static void main(String[] args) {
        int someNumber = 1;
        timesTwo(someNumber); // line 4
        System.out.println(someNumber);
    }
    static int timesTwo(int someNumber) {
        someNumber = someNumber*2;
        return someNumber;
    }
}
```

16. Write two different statements to print the last element of the array declared with the following declaration.

```
int[] x = new int[10];
```

This is a closed notes, closed book practice midterm.
... Actually you can do what you want, it's for your practice. I just recommend that.

17. Which answer below best describes this code fragment:

```
int[] x = new int[10];
for (int i = 0; i <= 10; i++)
    x[i] = i;
```

- The fragment will cause the program to crash with a NullPointerException.
- The fragment will cause the program to crash with an ArrayIndexOutOfBoundsException.
- The fragment will complete normally storing the values 0, 1, 2, etc. into the corresponding elements of the array.
- The fragment will cause the compiler to generate an error message.

18. What does the following program print? _____

```
class Question18 {
    public static void main(String[] args) {
        int[] array = {56, 6, 96, 10, 27, 45, 100, 14, 29};
        System.out.println(what(array, 2, 6));
    }
    static int what(int[] data, int start, int end) {
        int x = data[start]; // initial guess

        for (int i = start+1; i <= end; i++)
            if (data[i] < x)
                x = data[i];
        return x;
    }
}
```

19. Write a **function/method** with a single integer parameter, height, that prints a triangle of *'s like the one below. As an example, the one below is the result of calling the function with the value 5 (e.g. drawTriangle(5)). Your function must work for any value of the parameter height greater than 0. You may of course write helper functions as needed. Do NOT write a complete program; just provide the drawTriangle() function definition and the definitions of any additional functions needed by drawTriangle(), if any. The function does NOT read anything from the console (i.e. don't call in.nextInt()).

```
*****
****
***
**
*
```

This is a closed notes, closed book practice midterm.
... Actually you can do what you want, it's for your practice. I just recommend that.

20. Write a **function**, `vectorAppend`, that takes two parameters, each an array of integers. The function should return a new array that is the result of appending the contents of the second array parameter onto the end of the contents of the first. For example, if the first parameter was an array with the elements 10, 20, 30, and the second parameter was an array with the elements 15, 16, then the resulting array would have 5 elements: 10, 20, 30, 15, and 16. **DO NOT** write a complete program, write only the function.

21. Write a program that reads in an integer, and then creates a square multiplication table in a two-dimensional array of integers from 1 up to that number. For example, if you read in 3, then you create a 3 x 3 two-dimensional array that stores in row 0: 1x1, 2x1, 3x1; row 1: 2x1, 2x2, 2x3; row 2: 3x1, 3x2, 3x3. For extra practice, write a function that prints out the array, and call it from `main()`.

```
1 2 3
2 4 6
3 6 9
```

```
class MultiplicationTable {
    public static void main (String[] args) {
        Scanner in = new Scanner (System.in);
```

```
}
```

This is a closed notes, closed book practice midterm.
... Actually you can do what you want, it's for your practice. I just recommend that.

22. In the following program, there are a lot of syntax and runtime errors. This program is supposed to be a bubble sort: It goes through each element in the array from top to bottom, looking for the largest element. If the current element is larger than the one it is checking, it swaps them, thereby moving it closer to the bottom. Then the program decreases the indexes it needs to check (since the largest is on the bottom), and repeats until sorted. See if you can discover and correct all of them.

```
class Question22 {
    public static int main (String[] args) {
        Scanner input = new Scanner(System.in);
        int[] array = {7 54 29 2 98 184 46 21 59 0};

        bubblesort(array);
        for (i = 0; i <= array.length(); i++) {
            System.out.print (array[i])
        }

        void bubblesort(int array) {
            int temp, j;
            int search = array.length();
            for (int j = 1; j < array.length(); j++) {
                for (int i = 0; i < search; i++) {
                    if (array[i] > array[i+1]);
                    {
                        temp = array[i];
                        array[i] = array[i+1];
                        array[i+1] = temp;
                    }
                }
                search--;
            }
        }
    }
}
```

23. Write a method called `GuessingGame()` that takes in an integer. It uses a while loop to ask the user for a number over and over again until the user guesses his number correctly. Assume the number is from 1 to 100. Print out to the screen if the guess is too high or too low each time the user enters a guess.

Something to think about: In class, Professor Brandt went through an algorithm that would help finding an element in a sorted array. That same algorithm would work the same here in guessing the right number since the numbers are in the right order (hopefully...). What is that algorithm?

This is a closed notes, closed book practice midterm.
... Actually you can do what you want, it's for your practice. I just recommend that.

CMPS12A Study Guide Major Topics We've Covered

- ✓ When executing a program, the compiler immediately looks for public static void main(String [] args) to run.
- ✓ boolean expressions: these include
 - relational operators: <, <=, >, >=
 - equality operators: ==, !=
- ✓ if/else statements: either do it or don't or do something else
 - if (boolean expression) {
 - // do this if boolean expression evaluates to true
 - }
 - else {
 - // do this if boolean expression evaluates to false
 - }
 - You can have nested if/else statements, as well as if/else if/else statements. You can always use a switch/case statement if you have too many if/else statements, and are using an integer or char.
- ✓ blocks: When a variable is defined, it only stays defined for the block it is defined in. This is normally determined by the set of curly braces {}. Once the block ends, the variable ceases to exist.
- ✓ logical operators:
 - && AND (Both must be true)
 - || OR (Either is true)
 - !(boolean expression) NOT (True if boolean expression is false and vice versa)
- ✓ while loops: keep doing this until I say stop
 - while (boolean expression) {
 - // do while boolean expression is still true
 - }
 - Remember that you must update a variable in the boolean expression, or the loop will cycle forever.
- ✓ for loops: do the initialization once when loop hasn't started, run loop if boolean expression is true, update expression, repeat loop if boolean still true
 - for (initialization; boolean expression; update expression) {
 - // do this until boolean expression is false
 - }
 - Great for going through lists of things like arrays
- ✓ Methods/Functions
 - public static returnType identifier (parameter list)
 - public/private/default determine how visible function is to other functions outside the class
 - static allows other static methods to access it; if missing, then it is an instance method
 - returnType is the return type of the function. Could be any primitive or non-primitive type.
 - identifier is the name of the method
 - parameter list is what you are passing into the function. Remember to define what types they are.
 - Functions allow programs to access code more than once, and reduces complexity and length of the main function.
- ✓ Arrays
 - type [] name = new type [size];
 - come built in with a field .length that returns the length
 - count from zero to length -1 for the indexes. for loops are your friend.
 - You can also assign values by using { place values in here separated by commas }
 - Access fields by using name[index].
 - When declaring arrays, you declare a place in memory that will point to a list of whatever type the array is once you assign it. Thus, if you pass an array pointer to a method, you can still change the array because the pointer points to the same place in memory as the original pointer.
- ✓ 2D Arrays
 - Creates an array of references to arrays
 - type [] [] name = new type [size] [size2];

This is a closed notes, closed book practice midterm.

... Actually you can do what you want, it's for your practice. I just recommend that.

- ✓ comments: Two types of comments
 - `/* */` Mainly used for multi-line comments; anything inside these is treated as white space.
 - `//` Mainly used for a single line comment; anything after this on the same line is treated as white space.
- ✓ keywords – built in java words that are used for something in java. You cannot use these for an identifier. These are always lowercase. Example: break, continue, int, float, double, class
- ✓ identifiers – the names for variables used throughout a java program. These must start with a letter, `_`, or `$`, and can contain any of those and digits. Java style has uppercase letters to start new words. Example: x, nextInt, HelloWorld, WhatsUp1
- ✓ Type conversion
 - When performing an operation with a double and any other numeric type, it is converted to a double. Then float, then long, and finally, int.
 - If you want to force a conversion from double to int: `int i = (int)sum; // double sum;`
- ✓ Order of operations
 - Parenthesis are done first.
 - Multiplication and Division are done left to right.
 - Addition and Subtraction are done left to right.
 - Remember to initialize any variable before adding to itself (like ++)
- ✓ Method Overloading
 - You can create two methods with the same name, but taking different parameters with different types.
 - `static int min (double x, double y)`
 - `static int min (int x, int y)`
 - The compiler will automatically know which to use.
- ✓ Recursion
 - Recursion involves a method calling itself. You will need a base case for the recursion so that the method will eventually stop calling itself.
 - Each time the method calls itself, you will want the parameters passed in to change, which help calculate the answer.
 - Try tracing each call to the method separately to help see the result.
 - Don't expect to get it at once! Recursion is a difficult topic, and you will start to see more of it in CE16, CS12B, and CS101. Even I need to work at a recursion problem a couple times before I see the answer.
- ✓ Practice! Write more code. Use the book questions as a guide, and I can help you pick out good ones and check your answers. The key to being a good programmer is to practice above and beyond the assigned work. I know for sure that if you do the minimal amount of programming that is assigned in this class, you will most likely have a tough time when it comes to the exams.
- ✓ Email me: Matthew Low m1ow@ucsc.edu if you have any questions!
- ✓ MSI Sessions @ Baskin Whiteboards:
 - Tuesday 2:00-3:15 PM (2/21)
 - Wednesday 2:00-3:10 PM (2/22)
 - Monday 5:00-6:15 PM (2/27)
 - Make-up Session TBA
 - Midterm 2 Review Session TBA