This handout is meant as a review for the final exam in CS124. Questions 1 through 9 are a complete final exam from a previous semester. I added a few questions from other exams. This is not meant to cover every possible topic that might be on the final this year, but it will give you some idea of what types of questions might be.

---

1. Some short essay questions…

a) The central concept in computer science is *algorithm*. Define the term algorithm and explain the difference between an algorithm and a program.

b) Discuss *exceptions*—what they are and how and why they are used in Java.

c) What are *pointers*, and what are some of the consequences of the fact that variables in Java hold pointers to objects, rather than objects themselves?

d) Explain the meaning of the special variables *this* and *super*, and indicate some of the ways they can be used.

2. Some shorter programming problems…

a) Write a code segment that reads two integers from the user and prints their sum. Use *TextIO* for input. Include variable declarations.

b) Use a *while* loop to simulate flipping a coin until the coin has come up heads one thousand times. At the end, output the number of times that the coin has been flipped.

c) Suppose that A is an array of *int* that has already been filled with data. Write a code segment that calculates and outputs both the number of even integers in A and the number of odd integers in A.

3. Write a subroutine named *containsAll* with return type *boolean* and two parameters of type *String*. The value of *containsAll*(*str, chars*) should be *true* if the string *str* contains **every** character in the string *chars*. (You will need to use either nested *for* loops or the string method *indexOf*.)

4. Suppose that the two-dimensional array *rainfall* has been created as

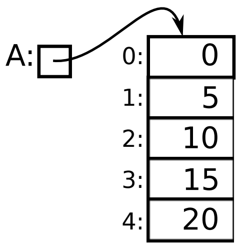
```
double[][] rainfall = new double[100][12];
```

and that *rainfall* has already been filled with data about rainfall in Geneva, NY, in the 20th century. In particular, *rainfall*$[y][m]$ is the rainfall, in inches, in the year $2000 + y$ for the m^{th} month of the year.

a) Write a code segment to compute and print the *average* April rainfall. (That is, find the average of the 100 values for month number 3 in all the years.)

b) Write a code segment to print the *total* rainfall in *each* of the 100 years. You should label the output with the year; for example, 1953: 37.2 inches

5. Write a code segment to create the situation shown in the picture, including an array of type *int*`[]` and a variable A that points to the array. (Include the variable declaration.)



6. A class named *Item* represents an item for sale on some web site. It has an instance method named *getCost*(), with no parameters, that returns a *double* giving the cost of the item. Write a subclass of *Item* that adds an instance method *getCostWithTax*() that returns the cost of the item plus a 7% tax. (You will have to use *super*!)

7. **a)** Write a simple class to represent a *bank account*. The class should have a constructor to create an account containing a given amount of money. And it should have three methods: to *deposit* money into the account, to *withdraw* money from the account, and to *get* the current amount of money.

b) Using your bank account class, write a code segment that creates an object representing an account containing $1000.00, then uses instance methods to deposit $159.27 into the account and print the current amount in the account.

8. Here are two methods that produce the same answer:

```
public static boolean test1( int[] A ) {
    int i,j;
    for (i = 0; i < A.length; i++) {
        for (j = i+1; j < A.length; j++) {
            if ( A[i] == A[j] ) {
                return true;
            }
        }
    }
    return false;
}
```

```
public static boolean test2( int[] A ) {
    int i;
    Arrays.sort(A);
    for (i = 1; i < A.length; i++) {
        if ( A[i-1] == A[i] ) {
            return true;
        }
    }
    return false;
}
```

a) What is meaning of the answer returned by these two methods? Explain briefly in words the process that each method uses to compute the answer.

b) Given that the method *Arrays.sort* uses a fast and efficient sorting algorithm, which of these methods is faster? Why?

9. Developing complex programs to solve complex problems is a fundamental part of computer science. Discuss some of the approaches that you, as a programmer, can apply when designing and writing computer programs. Also describe some features of Java—such as control structures, subroutines, objects, and events—that you can use as you design your programs. Specific examples will be useful.

---

10. Show the output of each of the following program segments:

a)
```
int x,y;
x = 100;
y = 0;
while (x > 0) {
  x = x / 2;
  System.out.println(x);
  y++;
}
System.out.println(y);
```

b)
```
int[] A,B;
A = new int[5];
B = new int[5];
A[0] = 1;
B[0] = 0;
for (int i = 1; i < 5; i++) {
  A[i] = 2 * A[i-1] + 1;
  B[i] = A[i] + B[i-1];
  System.out.prilnln(A[i] + " " + B[i]);
}
```

11. What do you suppose would be the purpose of the following method? Why?

```
public void actionPerformed(ActionEvent evt) {
    String command = evt.getActionCommand();
    if (command.equals("Start")) {
        startAnimation();
        button.setText("Stop");
    }
    else if (command.equals("Stop")) {
        stopAnimation();
        button.setText("Start");
    }
}
```

12. We have used the expression `(int)(1+6*Math.random())` to simulate the rolling of a standard 6-sided die, but some games use dice with different numbers of sides.

a) Write a complete Java class that represents a *single die* with any given number of sides. The class should have a constructor with no parameters that creates a standard 6-sided die. It should also have a constructor with one parameter of type *int* that specifies the number of sides of the die; the value of the parameter must be greater than 1. It should have a method for rolling the die and a method for reading the number that is currently showing on the die.

b) A certain game uses five dice, which have 4, 6, 8, 12, and 20 sides respectively. Write Java code that creates five objects belonging to the class from part **a)** to represent these five dice. You can use either five separate variables or an array to hold the data.

c) Write Java code that will roll the five dice that you created in part **b)** and print the sum of the numbers showing on the five dice.