1. Provide short answers (at most two sentences each) to the following questions.

   1.a What are the two components that a recursive method must have?
   
   A: A base case and at least one call to itself.

   1.b What is a necessary condition for a class to have multiple constructors?
   
   A: A class can have multiple constructors only if each constructor has a different set of parameters.

   1.c What is the scope of a variable?
   
   A: It is the subset of the program in which we can access the variable. It goes from the variable declaration to the closing brace corresponding to the opening brace above the variable declaration.

   1.d What are instance variables?
   
   A: Instance variables are variables that are associated to each instance of a class, i.e., to each object of that type. They define the properties of that specific object. They are initialized upon construction of the object, usually by the constructor.

2. Consider some program that needs to work with sets of values. Specifically, a set is a collection of unique values (no duplicates) for which the order is irrelevant. Assume to have access to the following methods for managing and manipulating sets of integers that are stored in arrays (you do not need to write these methods):

   - public static boolean contains(int[] set, int value):
     Return whether the given value is contained in the given set.

   - public static int[] copy(int[] set):
     Creates and returns a duplicate of the given set.

   - public static int[] insert(int[] set, int value):
     If the given value is not already contained in the given set, then return a new set with value inserted in it. If value is already present, then return the existing set, unmodified. Note that set may be null, in which case an array of length 1 containing value is returned.

   - public static int[] remove(int[] set, int value):
     If the given value is contained in the given set, then return a new set without value. If set had length 1 and contains value, return null, rather than an array of length 0. If value is not present, then return the existing set, unmodified.

To complete this collection of methods for use on sets, write the following two methods. You may use the methods listed above to write these two methods. You may assume that the arrays passed as parameters are not null.

---

1Although, for the sake of this practice, you may want to write them.
- public static int[] intersection(int[] setA, int[] setB):
  Return a new set that is the intersection of setA and setB. That is, return a set that contains only the values that appear both in setA and setB. If there are no elements in common, return null.

- public static int[] union(int[] setA, int[] setB):
  Return a new set that is the union of setA and setB. That is, return a set that contains the values that appear in either or both of setA and setB.

A:

```java
public static int[] intersection(int[] setA, int[] setB) {
    int[] res = null;
    for (int i = 0; i < setB.length; i++) {
        if (contains(setA, setB[i])) {
            res = insert(res, setB[i]);
        }
    }
    return res;
}

public static int[] union(int[] setA, int[] setB) {
    int[] res = copy(setA);
    for (int i = 0; i < setB.length; i++) {
        res = insert(res, setB[i]);
    }
    return res;
}
```

3. Consider Pascal’s triangle, show here in a usefully-loopsided form

<table>
<thead>
<tr>
<th>col</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

Specifically, we can define the value of any position in the triangle at row \( r \) and column \( c \) as

\[
T(r, c) = \begin{cases} 
1 & \text{if } c = 0 \text{ or } c = r \\
T(r-1, c-1) + T(r-1, c) & \text{if } 0 < c < r
\end{cases}
\]
That is, the left- and -rightmost columns of each row are always 1, while each “interior” value is the sum of the value above-and-to-the-left of it and the value above it.

Write a method with signature

```java
public static int[][] pascal(int r)
```

that returns an array-of-arrays of integers representing the first \( r \) rows of Pascal’s triangle. Each row is one element longer than the previous one, and so too should be the second-dimension arrays of the array-of-arrays.

A: A recursive solution is possible, but the following is an easier non-recursive solution.

```java
public static int[][] pascal(int r) {
    int[][] t = new int[r][];
    t[0] = new int[1]{1};
    for (int i = 1; i < t.length; i++) {
        t[i] = new int[i+1];
        t[i][0] = 1;
        for (int j = 1; j < t[i].length - 1; j++) {
            t[i][j] = t[i-1][j-1] + t[i-1][j];
        }
        t[i][t[i].length - 1] = 1;
    }
    return t;
}
```

4. You are trying to break into ACData, specifically the Registrar’s account, so that you can change your grades before your transcript gets sent to a bunch of potential employers.\(^2\) You do not know what the Registrar’s password is, but you can write code that searches for the right one.

The Registrar’s password cannot be more than 16 characters long, and can only contain alphanumeric characters (uppercase or lowercase letters, and the ten digits). Write a method crack() that attempts every possible 16-characters password of alphanumeric characters. Your crack() method should the return the correct password when it finds it.

To determine whether a given password is correct, your code can call the following, already written, method that returns whether the given password is the correct one:

```java
public static boolean isCorrect(char[] password)
```

You may also assume that the following array, which contains all 62 alphanumeric characters, is defined as a global variable, thus available for your method:

```java
char[] an = { 'A', 'B', , ... , 'Z', 'a', 'b', ... , 'z', '0', '1', ... , '9'};
```

\(^2\)Bad, bad, bad idea.
Finally, you may assume the existence of the following helper methods that perform simple operations on arrays:

- public static char[] copy(char[] a) returns a duplicate of the given char array a;
- public static char[] append(char[] a, char c) a new array of chars of length one more than the array a and contents equal to a in all-but-the-last positions, and c in the last position.

**Hint:** You likely want to use recursion here.

A:

```java
class PasswordCracker {
    public static char[] crack() {
        char[] password = new char[1];
        char[] pw = null;
        boolean found = false;
        for (int i = 0; i < an.length && ! found; i++) {
            password[0] = an[i];
            pw = crackHelper(password, 16);
            if (pw != null) {
                found = true;
            }
        }
        return pw;
    }

class CrackHelper {
    public static char[] crackHelper(char[] password, int maxLength) {
        if (isCorrect(password)) {
            return password;
        } else {
            if (password.length < maxLength) {
                char[] prefix = copy(password);
                char[] pw = null;
                boolean found = false;
                for (int i = 0; i < an.length && ! found; i++) {
                    pw = crackHelper(append(prefix, an[i]), maxLength);
                    if (pw != null) {
                        found = true;
                    }
                }
                return pw;
            } else {
                return null;
            }
        }
    }
}
```