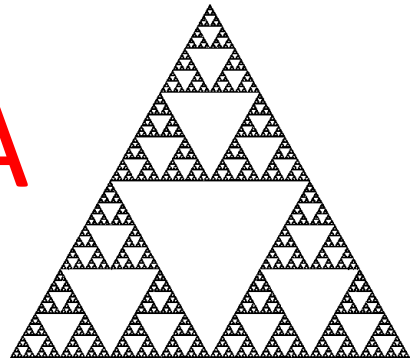
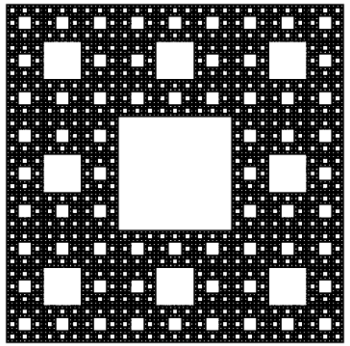
# Review

A bunch of syntax

A

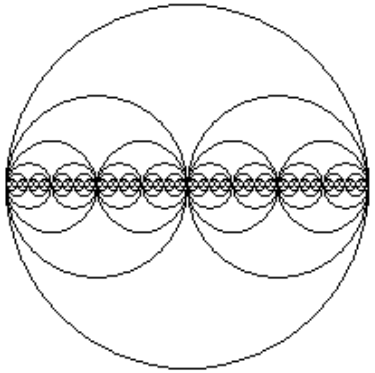


B

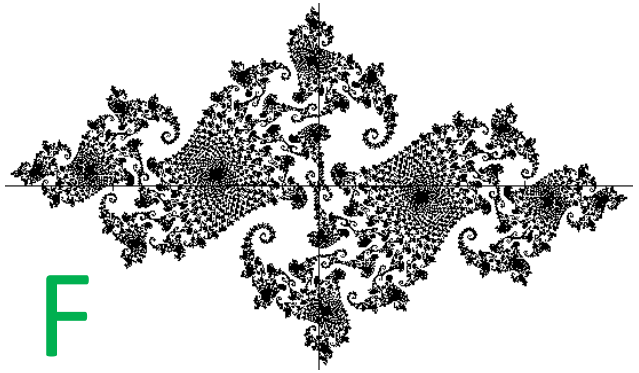


C

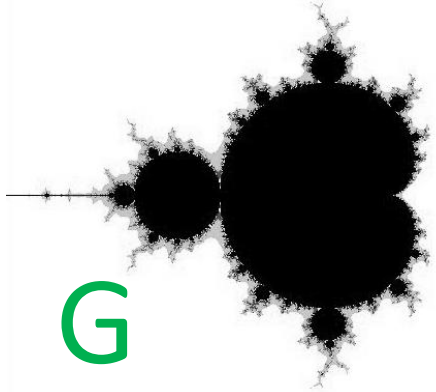
D



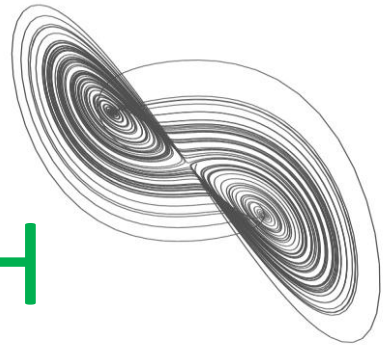
E



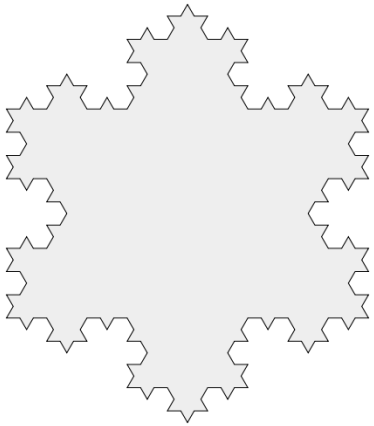
F



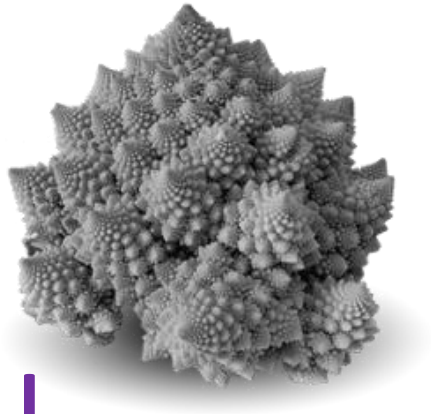
G



H

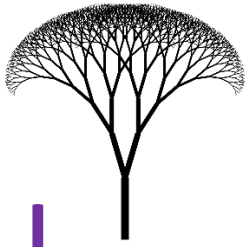


I



J

K



L

M

1. toLowerCase
2. toUpperCase
3. Length

4. charAt
5. indexOf
6. Substring

7. Replace
8. equals
9. compareTo

```
//Complex method
public returnType methodName (paramType paramName) {
    code;
    return something;
}
```

```
public int fib (int n) {
  if (n <= 2)
    return 1;
  else
    return fib (n - 1) + fib (n - 2);
}
```

```
//25 24 23 22 21 20 19 18 17

  public int seq (int n)
  {
      if (n <= 1)
          return [start num];
      else
          return seq (n - 1) – [change by];
  }
```

# Looping through a String

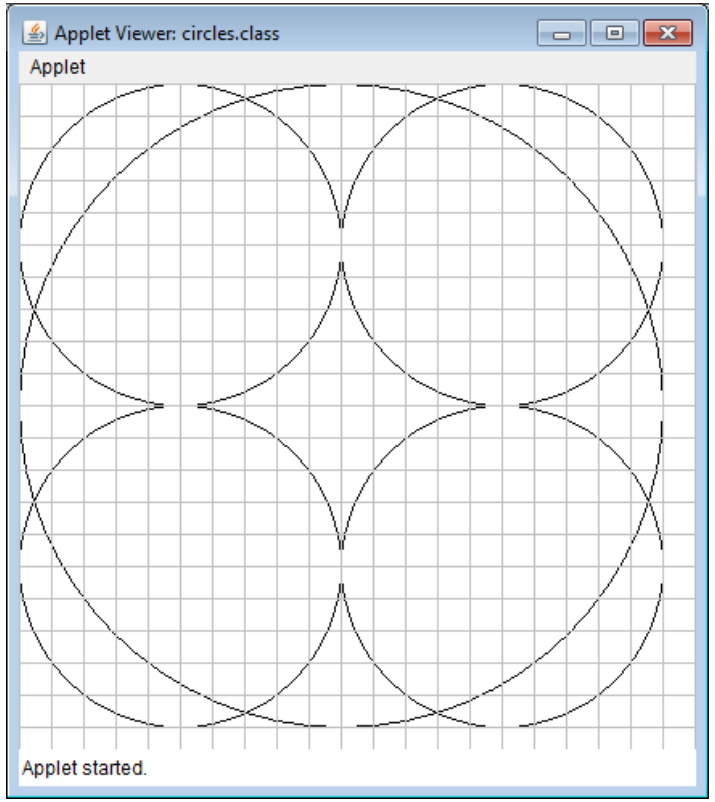
```
for(int i=0; i<s.length(); i++){
… s.charAt(i)…
}
```

Get the String, add “ay” to the end.
Print it out.

Get the String, print out the first and last letter.

Get the String, print out the second half of the word.

# Get the String. Print every second character.

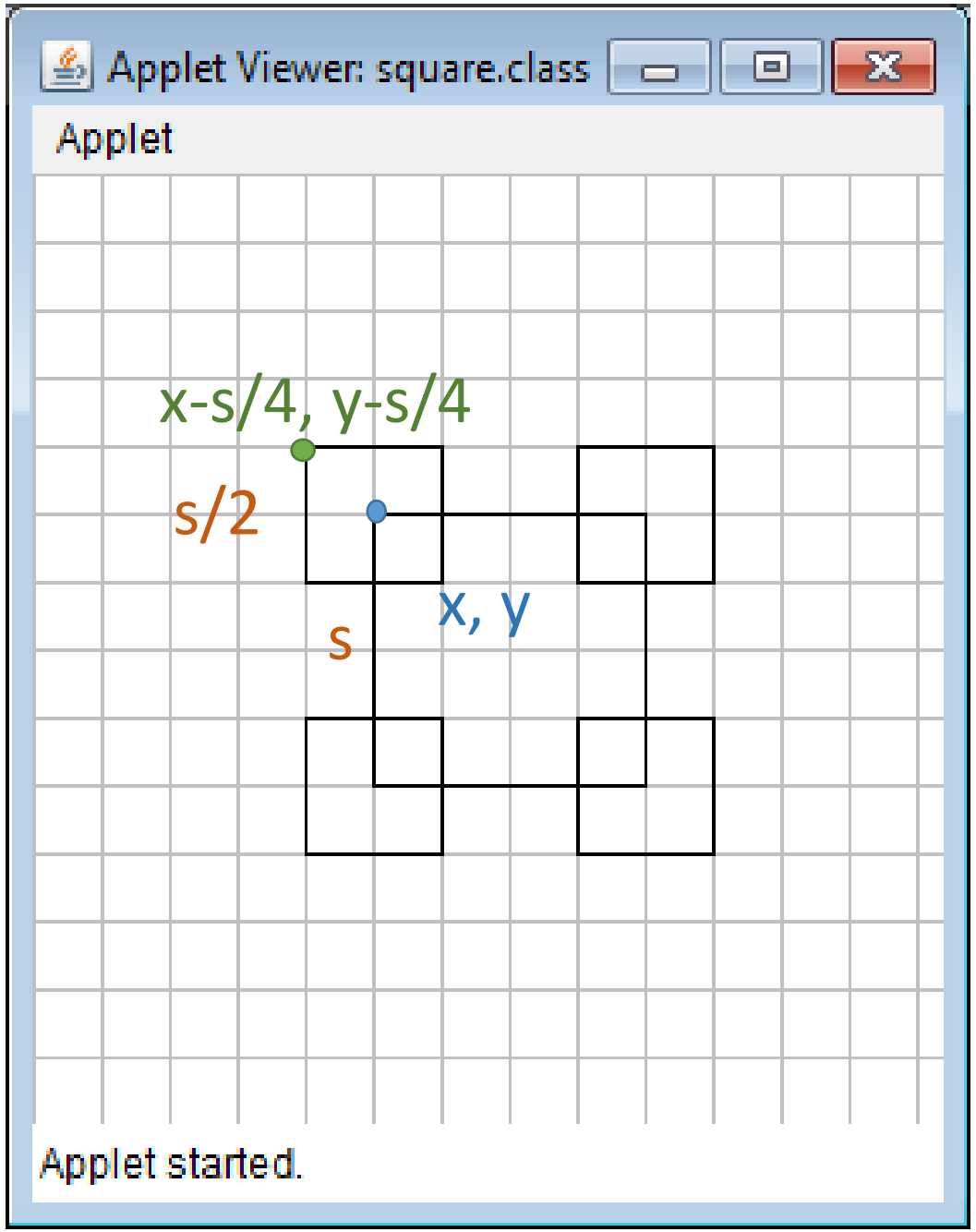



```
public void pattern (int x, int y, int d) {
  Graphics g = getGraphics ();
  if (d < 30)
     return;
  else {


  }
}
```

Step 1: Draw the big circle in terms of x, y, and d.

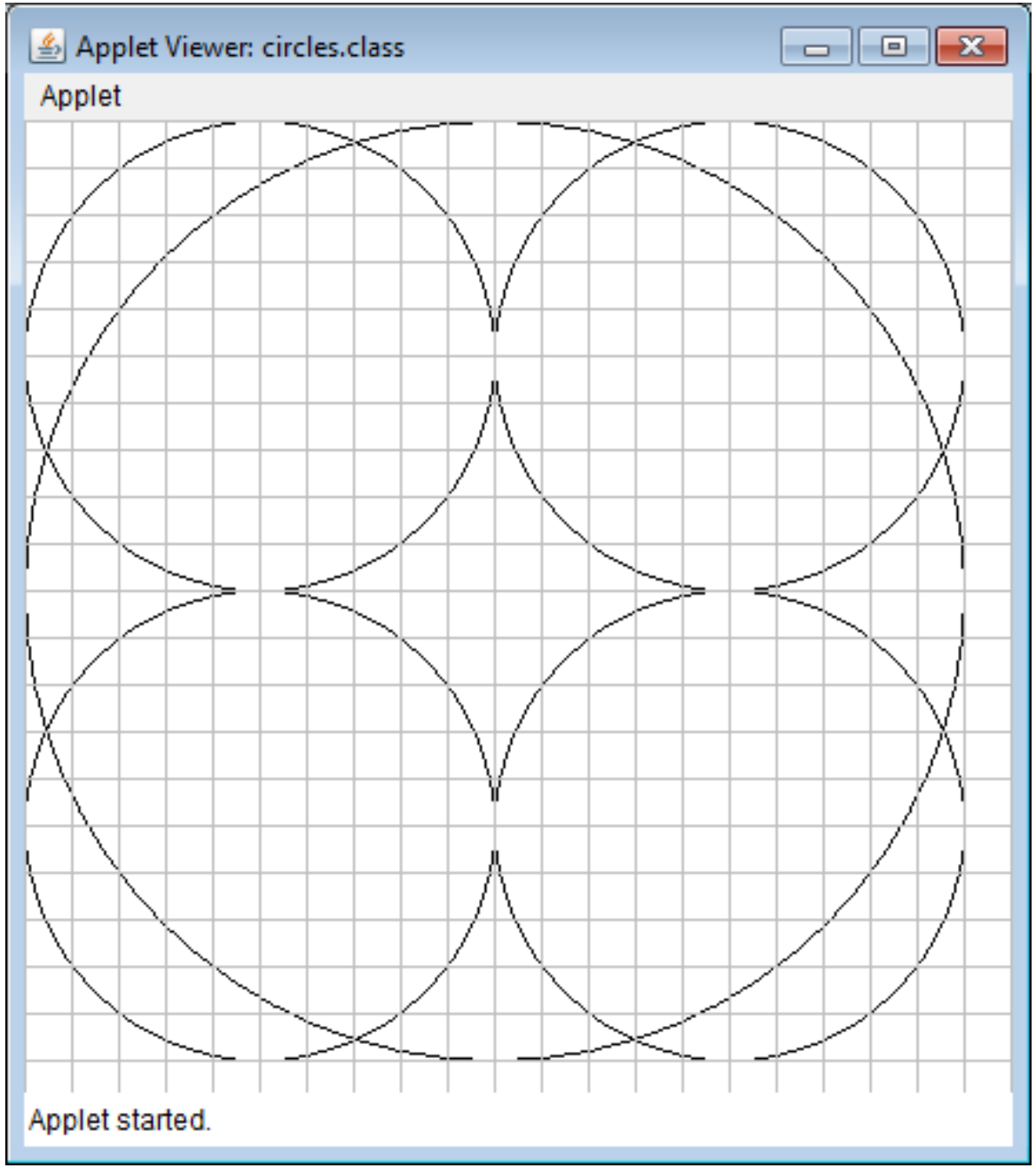
Step 2: Find the top-left co-ordinate and the side length of the other four shapes in terms of x, y, and d. Recursively call the method.

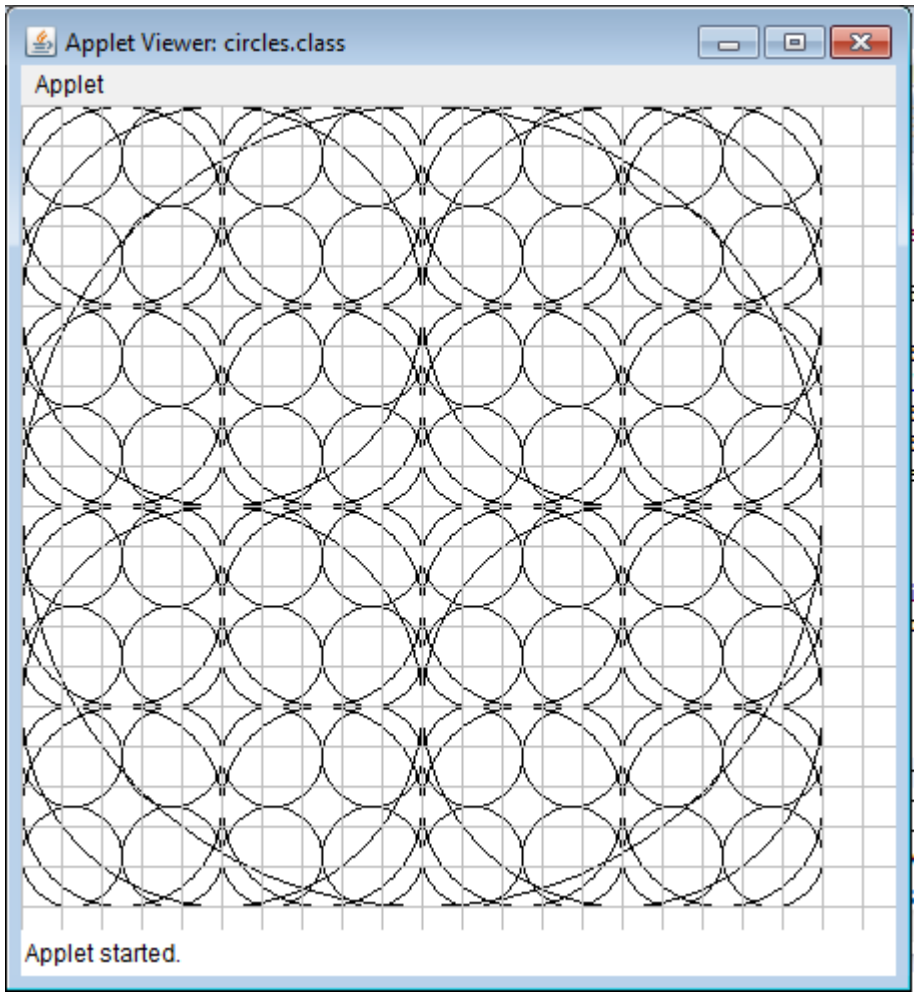



```
public void sq (int x, int y, int s) {
  Graphics g = getGraphics ();
  if (s <5)
     return;
  else {
     g.fillRect (x, y, s, s);
     sq (x-s/4 , y-s/4, s / 2);


   }
}
```

In terms of x, y, and s, what are the top left co-ordinate of the other 3 rectangles?


Applet Viewer: circles.class
Applet
Applet started.


Applet Viewer: circles.class
Applet
Applet started.

2. Use this code to fill in the memory diagram. Then, write out what each substring prints. /5

```
String at = "Air Traffic Controller";
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | |

| | | |
|---|---|---|
| (a) | `System.out.println(at.substring(0, 3));` | |
| (b) | `System.out.println(at.substring(4, 11));` | |
| (c) | `System.out.println(at.substring(2, 7));` | |
| (d) | `System.out.println (at.substring (at.length()-2, at.length()));` | |

3. Fill in the String memory diagram and then match the code with the output.

String gps = "Global Positioning System";

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | | | | | | | |

| | | | |
|---|---|---|---|
| | l | | a. System.out.println (gps.length ()); |
| | 3 | | b. System.out.println (gps.toUpperCase ()); |
| | 25 | | c. System.out.println (gps.toLowerCase ()); |
| | true | | d. System.out.println (gps.charAt (5)); |
| | false | | e. System.out.println (gps.equals ("radar")); |
| | Gljbal Pjsitijning System | | f. System.out.println (gps.compareTo ("radar") < 0); |
| | global positioning system | | g. System.out.println ((int) gps.charAt (0)); |
| | GLOBAL POSITIONING SYSTEM | | h. System.out.println (gps.indexOf ('b')); |
| | 71 | | i. System.out.println (gps.replace ('o', 'j')); |

15. Circle and correct an error in each line of code.

```
String a = "plane"

System.out.println (plane.length());

System.out.println (a.charAt (3) + " " + a.char (4));

System.out.println (a.subSTRING (3, 5));

System.shout.println (a.equals("jet"));
```

4. Fill in the String memory diagram and then match the code with the output

String air = "airCRAFT carrier";

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | |

| | | | |
|---|---|---|---|
| | AIRCRAFT CARRIER | | System.out.println (air.length ()); |
| | true | | System.out.println (air.toUpperCase ()); |
| | aircraft carrier | | System.out.println (air.toLowerCase ()); |
| | 14 | | System.out.println (air.charAt (5)); |
| | A | | System.out.println (air.equals ("windspeed")); |
| | aiyCRAFT cayyiey | | System.out.println (air.compareTo ("windspeed") < 0); |
| | false | | System.out.println (air.replace ('r', 'y')); |
| | 16 | | System.out.println (air.indexOf ('e')); |

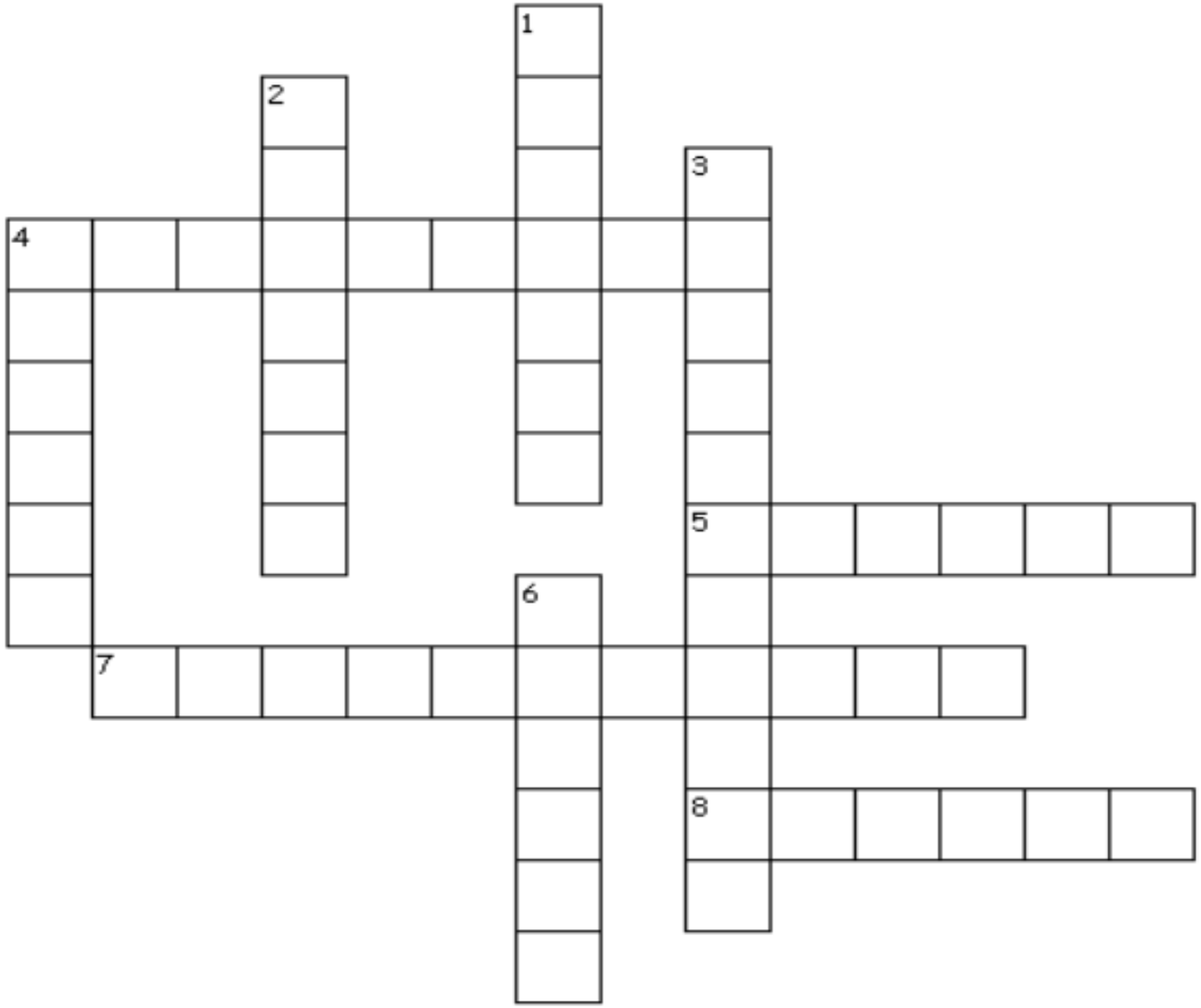
5. What would each of these lines of code print on the screen?

String word = "AiRpLaNe";

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

| | |
|---|---|
| | a. System.out.println (word.length ()); |
| | b. System.out.println (word.toUpperCase ()); |
| | c. System.out.println (word.toLowerCase ()); |
| | d. System.out.println (word.charAt (3)); |
| | e. System.out.println (word.charAt (5)); |
| | f. System.out.println (word.charAt (6)); |
| | g. System.out.println (word.indexOf ('R')); |
| | h. System.out.println (word.indexOf ('p')); |
| | i. System.out.println (word.indexOf ('e')); |

6. Complete the following word search.




*Across*

4. Tests if one String is greater than the other.
5. Tests if two Strings are the same.
7. Turns the String to capital letters.
8. A type that is a collection of characters that holds words.

*Down*

1. Finds a character in the letter and returns its index (or address).
2. Switches all of one letter in the String to a different letter.
3. Turns the String to small letters.
4. Picks a character out of the String from a specific position.
6. A number representing how long the String is.

# Phone Format

2.10 c

Name: _____

- In this app, a 10 character phrase is inputted – the user is supposed to enter a phone number, but they may not.
- Then, if there are 10 characters it is reformatted to this style: (XXX) XXX-XXXX
- The reformatted output, or an error message, is outputted.

**First, understand the input & output:**

Enter a phone number:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

The same number, reformatted:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | |

**Second, code it:**

```
//Make the onClick method



//Find the widgets that change (lost due to inflation)



//Get the phone number



//Pull out the pieces you need...substring!
//Store in variables.



//Write the if and update the screen
```

```
//Make the onClick method

public void answer(View view){

//Find the widgets that change (lost due to inflation)

TextView result =(TextView) findViewByID(R.id.result);

EditText phone =(EditText) findViewByID(R.id.phone);


//Get the phone number

String n = phone.getText().toString();

//Find the 4th, 5th, 6th and 7th digits.
//Store in variables.

char n4=n.charAt(3);      char n6=n.charAt(5);
char n5=n.charAt(4);      char n7=n.charAt(6);


//Write the if and update the screen

if ((n4=='8' || n4== '9') && (n7=='8' || n7=='9') && n5==n6)
   result.setText("Ignore");
else
   result.setText("Answer");
}
```

# Telemarketer or Not?

2.11 c

Name: _____

Here at the Concerned Citizens of Commerce (CCC), we have noted that telemarketers like to use seven-digit phone numbers where the last four digits have three properties. Looking just at the last four digits, these properties are:

- the first of these four digits is an 8 or 9;
- the last digit is an 8 or 9;
- the second and third digits are the same.

For example, if the last four digits of the telephone number are 8229, 8338, or 9008, these are telemarketer numbers.

Write a program to decide if a telephone number is a telemarketer number or not, based on the last four digits. Only 7 digit numbers will be entered. If the number is not a telemarketer number, we should answer the phone, and otherwise, we should ignore it.

```
//Make the onClick method

//Find the widgets that change (lost due to inflation)

//Get the number and the character
```

String n = input.getText().toString();

```
//Find the number and the character.
//Store each in variables.
```

String num =n.substring(0, n.indexOf(‘ ‘));
int number = Integer.parseInt(num);

```
//Update the screen, use a loop
```

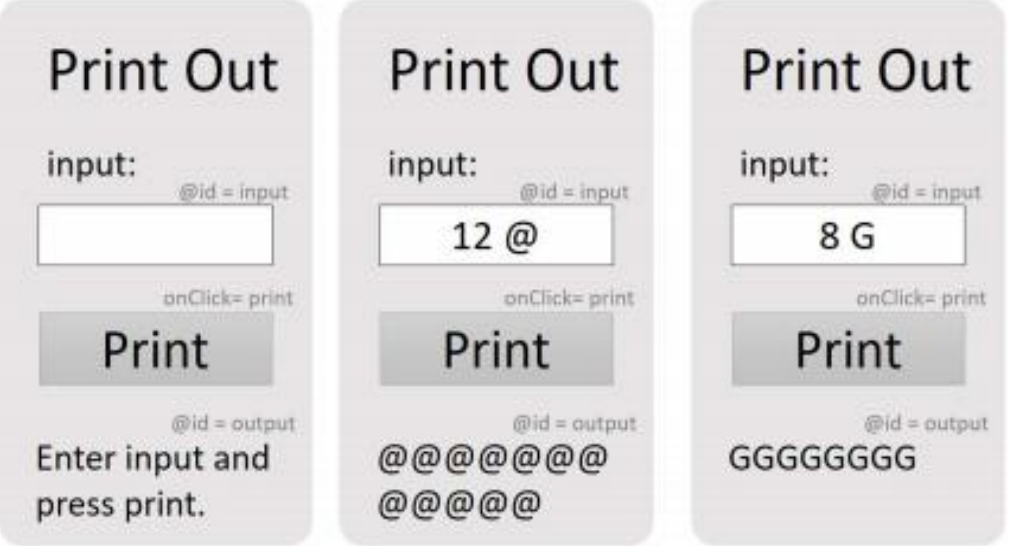
# Print Out N Times

2.12

Name: _____

Each input contains one positive integer less than 80, followed by one space, followed by a (non-space) character. The character is printed out the number of times indicated by the positive integer.

You may assume that all input follows this format.




```
//First, understand the two pieces of input:
```

12 @

| 0 | 1 | 2 | 3 |
|---|---|---|---|
| | | | |

8 G

| 0 | 1 | 2 |
|---|---|---|
| | | |

```
//Make the onClick method

//Find the widgets that change (lost due to inflation)

//Get the word

//Make a new blank String to hold the word

//Write the loop to reverse the String

//Update the screen
```

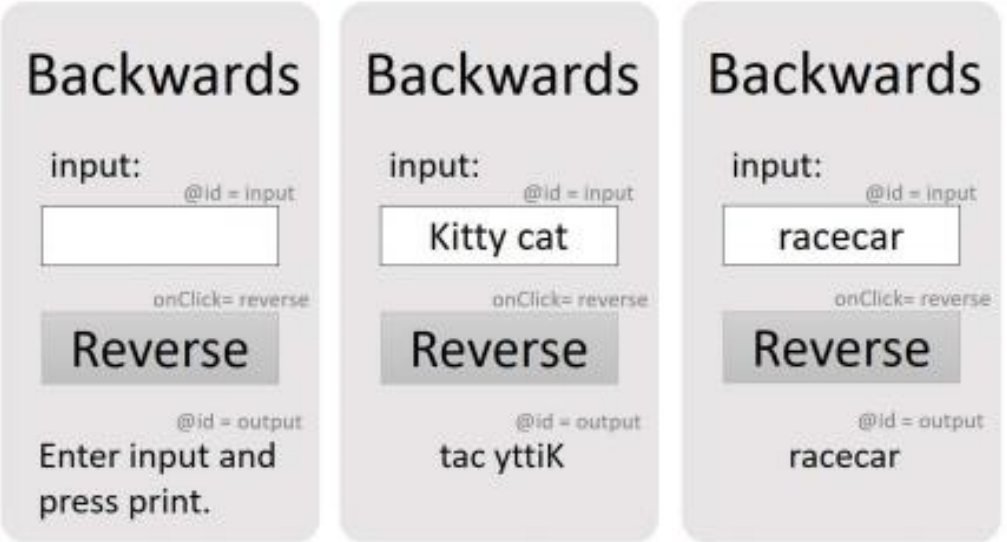
```
String temp="";
For(int i=s.length()-1; i>=0; i--)
    temp+=s.charAt(i);
```

# Reverse Word

2.12

Name: _____

- In this app, a word or series of words, of any length is entered.
- The program reverses the word and prints it on the screen.




```
//First understand what is printed:

Kitty cat:
```

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |

What is the first letter printed? . . . . . .

What is the first printed letter's position? . . . . . .

Is it always in that position for all words? YES or NO (circle)

Write the first printed letter's position using s.length(): . . . . . .

What is the last letter printed? . . . . . .

What is the last printed letter's position? . . . . . .

Is it always in that position for all words? YES or NO (circle)