PLEASE NOTE THESE SOLUTIONS HAVE NOT BEEN CHECKED FOR CORRECTNESS AND MAY CONTAIN A TYPO OR OTHER ERROR.

1. Short Answers

(a) What is printed by the last two lines of the following Java code? Assume that the constructor initializes the private instance variables and that no compile or runtime errors occur. You do not need to write an explanation.

Write exactly the program output here and nothing else:YS

(b) Briefly explain why no String objects are created in the following code.

An array of String pointers (references) are created.

(c) Decide if a class method, instance method or constructor is being used. Circle the correct response.

`File f  = new File("a.txt");`	{ class , instance , **constructor** }
`int m   = Math.max(i,j);`	{ **class** , instance , constructor }
`String s = "a".toUpperCase();`	{ class , **instance** , constructor }
`TextIO.putln(42);`	{ **class** , instance , constructor }

(d) Which variable in the class below is *not* stored as part of each Corn object? **warm**

```
public class Corn { // Your code will directly use lines 2-6
2  public Corn(int initial) { height = initial; }
3  public static boolean warm = true;
4  public int height;
5  public void grow() { if(warm) height+=2; else height +=1; }
6  public boolean isReady() { return height > 5;}
}
```

(e) Your out-of-state friend incorrectly claims Corn objects are immutable.
i) Complete the main method below to call or directly use all of Corn's public methods and variables.
ii) Precisely circle two parts of your code to illustrate two different reasons why your friend is wrong about Corn.

```
public static void main(String[] arg) {
  Corn c = new Corn(1);
  Corn.warm = false;
  [c.grow();]
  [c.height = 10;]
  c.isReady();
```

2. Color by name. – 16 points

You're creating a new app. Complete the following methods below for the `Map` class.

```
public class Pair {
  public String key;
  public Color value;
}// Pair has no constructors!
```

```
public class Map {
  private Pair[] array = new Pair[0]; // The Pair class is defined here ..
```

/* *Adds the given name and color pair to the map. Assume the name is non-*`null` *and has not been added before. Assume color is non-*`null` */

```
  public void add(String name, Color color) {
    // Write code below to create a Pair object to hold the given name and color.
    Pair p = new Pair();
    p.key = name;
    p.value = color;
    // Complete the code below add your new pair object to the map's array.
    Pair[] temp = new Pair[array.length+1];

    for(int i =0; i < array.length ; i++)

        temp[i] = array[i];

    temp[ temp.length-1] = p;

    array = temp;
}
```

/* *Returns the color object previously* added *to this map for the given name. Returns* `null` *if no such entry exists. Hint: Do not compare strings using* == */

```
  public Color find(String name) {
     Pair p = null;
     for(int i =0; i < array.length; i++) {
         if(array[i].key.equals(name))
             return array[i].value;
     }
     return null;
  }
```

/ Resets the map to be empty again. */*

```
public void reset() {
     array = new Pair[0];
  }
```

/ By using the above* `find` *method write minimal code (e.g. 1 line): Return* `true` *if the map contains the given key,* `false` *otherwise. */*

```
public boolean contains(String name) {
     return find(name) != null;
  }
} // end class
```

3. I took CS125 and all I got was this startup company and nerdy friends

(a) Write code below so that `mosaic` returns an image, represented as a 2D array which is *the same size* as the original *input* array.

```
public static int[][] mosaic(int[][]input, int repeat) {
// Note the conflict problem is slightly different
if(input == null) return new int[1][1];
int width = input.length, height = input[0].length;
int[][] result = new int[width][ height];
for(int x = 0;x< width;x++)
  for(int y =0; y < height;y++) {
     int srcX = (x*repeat) % width;
     int srcY = (y*repeat) % height;
     result[x][y] = input[srcX][srcY];
  }
return result;
}
```

(b) Complete the description of the following image transformation. (Note conflict may be different)

"For each pixel, when the ***BLUE*** component is greater than 200,
the output is ***GREEN*** otherwise it is ***White*** }"

(c) Write an expression that returns true if *pixel1*'s red component is greater than the *pixel2*'s blue component.

```
public static boolean test(int pixel1, int pixel2) {
      return ((pixel1&0xff0000)>>16) > (pixel2 & 0xff); // assumes alpha component
      return (pixel1>>16)&0xff > (pixel2 & 0xff); // assumes alpha component
      return (pixel1>>16) > (pixel2 & 0xff); // assumes bits 24-31 are 0
      return pixel1 > ((pixel2&0xff) <<16); //bizarre but correct.
}
```

4. Tweet data mining.

```
public class Tweet {
      private String message, sender;
      private int count;
      public Tweet(String m, String s) {message = m; sender = s;}
      public Tweet(Tweet t) {message = t.message; sender= t.sender;count=t.count}
      public boolean retweet() { count++; return count >= 1000;}
      public boolean retweet() { return ++count >= 1000;}
      public boolean retweet() { return count++ >= 999;}
      public boolean isBig(Tweet t) { return t==null || this.count >t.count;}
}
```

5. Class methods are static?

```
public boolean hasNext() { // true if roll() can still return another value.
            return count < array.length;
```

// (ii) '*create*' that takes an integer parameter 'n' and returns array of *n* initialized Dice objects,

```
    public static Dice[] create(int n) {
        Dice[] array = new Dice[n];
        for(int i =0; i < n;i++)
            array[i] = new Dice();
         return array;
    }
```

//(iii) '*deepcopy*' that takes and returns an array of Dice objects. Use the Dice copy constructor to return a deep copy of the given array. Do not modify the original array.

```
    public static Dice[] deepcopy(Dice[] source) {
        Dice[] copy = new Dice[source.length];
        for(int i =0; i < n;i++)
           copy [i] = new Dice(source[i]);
        return copy;
    }
```

} In a different class, *DicePrint* use the Factory and Dice methods to complete the main method below: (vi) Create a deep copy of the given array. (v) Use *hasNext* and *roll* to print out all of the values of the first dice of the deep copy.

```
public class DicePrint {
  public static void print(Dice[] dice) {
   Dice[] mycopy = Factory.deepcopy(dice)
   Dice d = mycopy[0];
   while(d.hasNext())
      TextIO.putln(d.roll());
```

6. Concepts and code snippets – 13 Points (Last Question)

(a) The variable 's' refers to an empty stack, the variable 'q' refers to an empty queue. Determine the contents of the *result* array below.

__40__   __60__   __50__   _30__   _70__

(b) Write one line after the following code: Use variable '*ss*' to cause a *NullPointerException* to be thrown.

ss[0].toUpperCase();

(c)

A programmer wants to check the value of two variables while the program is executing using a debugger. By using **Breakpoint** the program execution can be paused at a specific line and the variable values can be inspected.

(d) Your friend has written the following buggy code. Rewrite the code below to return the count of true entries in the array. For example, for the array `{true,false,true}` the method should return 2. Hint: There are at least three lines of code to fix.

public static int count(boolean[] array) {
  for(int i =0; i <= array.length; i++) {
    int c = 0;
    if(array[i] = true) ;
      c = c + 1;
    else return c;
  }
}
	public static int count(boolean[] array) {
	  int c = 0;
	  for(int i =0; i <= array.length; i++) {
	    if(array[i])
	      c = c + 1;
	  }
	  return **c**;
	}