Interpretation Worksheet Answer Key

Problem 1.0. What does the following code print when the following input is used?

```
3
Hi There Friends!
```

```
1  public class Interpret01 {
2      private String [] input;
3
4      public Interpret01 (Scanner sc){
5          input = new String [sc.nextInt()];
6          for (int i = 0; i < input.length; i++)
7              input[i] = sc.next();
8          sc.close();
9      }
10
11     public char charAt (int aIndex, int sIndex){
12         return input[aIndex].charAt(sIndex);
13     }
14
15     public int [] indexOf (char c){
16         int [] indices = new int[2];
17         for (int a = 0; a < input.length; a++)
18             for (int s = 0; s < input[a].length(); s++)
19                 if (input[a].charAt(s) == c){
20                     indices[0] = a;
21                     indices[1] = s;
22
23                 }else{
24                     indices[0] = -1;
25                     indices[1] = -1
26                 }
27         return indices;
28     }
29
30     public static void main (String [] args){
31         Scanner scanner = new Scanner (System.in);
32         Interpret01 a = new Interpret01(scanner);
33         System.out.println(a.charAt(1, 1));
34         System.out.println(Arrays.toString(a.indexOf('z')));
35         System.out.println(Arrays.toString(a.indexOf('e')));
36     }
37 }
```

Problem 1.1. What does the following code print?

h
$[-1, -1]$
$[-1, -1]$
Notes: Because this is using Arrays.toString() to return the array, you need an opening square bracket, the contents (separated by commas) and a closing square bracket.

'e' returns $[-1, -1]$ because of the if/else clause. Work through the example and then look at a more correct version of indexOf on the next page.

Problem 1.2. In one sentence explain what the charAt method does:

I'm going to use more than one sentence ;).
The charAt method takes two integers and returns a character. The first int represents the index of the String array and the second int represents the index in the String. This method returns the character that corresponds to these indices.

Problem 1.3. In one sentence explain what exactly the indexOf method does (be careful, this is an incorrect implementation - what makes it incorrect?):

Again a few more sentences for this explanation.
The indexOf method takes a character and returns an array of two integers. The first int represents the index of the String array and the second int represents the index in the String. However since we are using if/else, the only time this method works is when the last character of the last String equivalent to our parameter.

Problem 2.0. Here is a different implementation of the index of method:

```
1     public int [] indexOf (char c){
2         int [] indices = {-1, -1};
3         for (int a = 0; a < input.length; a++)
4             for (int s = 0; s < input[a].length(); s++)
5                 if (input[a].charAt(s) == c){
6                     indices[0] = a;
7                     indices[1] = s;
8                     return indices;
9                 }
10        return indices;
11    }
```

Problem 2.1. Describe what this implementation does:

The indexOf method takes a character and returns an array of two integers. This array either represents the first occurrence of the character in the String array (The first integer corresponds to the index in the array, the second integer corresponds to the index in the String) or $[-1, -1]$ to represent that the character was not found in the String array.

Problem 2.2. How does this method change when I delete line number 8?

If the return statement is deleted, then the method will return the last occurrence instead of the first.

Problem 3.0.

```
public class ReferenceType {
    private int a = 10;
    private int [] b = {10, 20, 30 , 40};

    public int getA(){
        return a;
    }

    public int [] getB(){
        return b;
    }

    public void add10(int c, int [] d){
        a += 10;
        c += 10;
        for(int i = 0; i < b.length; i++){
            b[i] += 10;
        }
        for(int i = 0; i < d.length; i++){
            d[i] += 10;
        }

        public static void main(String [] args){
            int myLocalInt = 20;
            int [] myLocalArray = {20, 30, 40, 50};
            ReferenceType ref = new ReferenceType();
            ref.add10(myLocalInt, myLocalArray);
            System.out.println(myLocalInt);
            System.out.println(Arrays.toString(myLocalArray));
            System.out.println(ref.getA());
            System.out.println(ref.getB());
        }
    }
}
```

Problem 3.1. What is printed?

20
[30, 40, 50, 60]
20
[20, 30, 40, 50]

The only variable that does not change is the local Primitive integer.

Problem 3.2. Why do I use an object of type ReferenceType to print the methods in the main class?

Because my main method (the calling method) is static while the methods that are being called are non-static (they probably access the instance variables).

Problem 3.3. Explain why no parameters are used when ref is created:

Because we have not created a constructor. When we don't create a constructor, a default constructor with no parameters is used.

Problem 4.0. What does the following code print?

Arrays.sort(myArray) is a static method in the Arrays class that will sort your array from the lowest value (if it is sorting by characters, then it sorts based on ascii values). So the following class prints:
12
21
34
45
78
89

```
1  public class ArraySort {
2
3      public static void main(String[] args) {
4          int[] a = new int[]{45, 12, 78, 34, 89, 21};
5          Arrays.sort(a);
6          for( int element : a){
7               System.out.println(element);
8          }
9      }
10 }
```