

RECURSION (Basics) SOLUTIONS

Solution 1:

```
public class Solution {
   public static void allOccurences(int arr[], int key, int i) {
      if(i == arr.length) {
        return;
    }

    if(arr[i] == key) {
        System.out.print(i+" ");
    }
    allOccurences(arr, key, i+1);
}

public static void main(String[] args) {
    int arr[] = {3, 2, 4, 5, 6, 2, 7, 2, 2};
    int key = 2;
    allOccurences(arr, key, 0);
    System.out.println();
}
```

Solution 2:

```
public class Solution {
    static String digits[] = {"zero", "one", "two", "three", "four", "five", "six",
    "seven", "eight", "nine");

public static void printDigits(int number) {
    if(number == 0) {
        return;
    }

    int lastDigit = number%10;
        printDigits(number/10);
        System.out.print(digits[lastDigit]+" ");
    }

public static void main(String[] args) {
        printDigits(1234);
    }
}
```



```
System.out.println();
}
```

Solution 3:

```
public class Solution {
   public static int length(String str) {
      if(str.length() == 0) {
        return 0;
    }
   return length(str.substring(1)) + 1;
}

public static void main(String[] args) {
   String str = "abcde";
   System.out.println(length(str));
}
```

COLLEGE

Solution 4:

```
public class Solution {
   public static int countSubstrs(String str, int i, int j, int n) {
      if (n == 1) {
         return 1;
      }
      if (n <= 0) {
        return 0;
      }

   int res = countSubstrs(str, i + 1, j, n - 1) +
            countSubstrs(str, i, j - 1, n - 1) -
            countSubstrs(str, i + 1, j - 1, n - 2);

if (str.charAt(i) == str.charAt(j)) {
        res++;
    }
</pre>
```



```
return res;
}
public static void main(String[] args) {
   String str = "abcab";
   int n = str.length();
   System.out.print(countSubstrs(str, 0, n-1, n));
}
```

Solution 5:

```
public class Solution {
   public static void towerOfHanoi(int n, String src, String helper, String dest) {
      if (n == 1) {
        System.out.println("transfer disk " + n + " from " + src + " to " + dest);
      return;
   }

      //transfer top n-1 from src to helper using dest as 'helper'
      towerOfHanoi(n-1, src, dest, helper);
      //transfer nth from src to dest
      System.out.println("transfer disk " + n + " from " + src + " to " + helper);
      //transfer n-1 from helper to dest using src as 'helper'
      towerOfHanoi(n-1, helper, src, dest);
}

public static void main(String args[]) {
   int n = 4;
   towerOfHanoi(n, "A", "B", "C");
}
```

The Solution for this particular question has also been discussed here : https://www.voutube.com/watch?v=u-HqzqYe8KA

At timestamp: 00:05