

Recursion Class 2

Q1. Tower of Hanoi - Transfer n disks from source to destination over 3 towers.

```
public class Recursion2 {  
    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");  
    }  
}
```

Q2. Print a string in reverse.

```
public class Recursion2 {  
    public static String revString(String str) {  
        if(str.length() == 1) {  
            return str;  
        }  
  
        char currChar = str.charAt(0);  
        String nextString = revString(str.substring(1));  
        return nextString + currChar;  
    }  
    public static void main(String args[]) {
```

```

        String str = "abcd";
        String reversed = revString(str);
        System.out.println(reversed);
    }
}

```

Q3. Find the occurrence of the first and last occurrence of an element using recursion.

```

public class Recursion2 {
    public static int first = -1;
    public static int last = -1;

    public static void getIndices(String str, char el, int idx) {
        if(idx == str.length()) {
            return;
        }

        if(str.charAt(idx) == el) {
            if(first == -1) {
                first = idx;
            } else {
                last = idx;
            }
        }

        getIndices(str, el, idx+1);
    }

    public static void main(String args[]) {
        String str = "tabcdfghijakkk";
        char el = 'a';
        getIndices(str, el, 0);
        System.out.println("First occurrence : " + first);
        System.out.println("Last occurrence : " + last);
    }
}

```

Q4. Check if an array is sorted (strictly increasing). - $O(n)$

```

public class Recursion2 {
    public static boolean checkIfIncreasing(int arr[], int idx) {

```

```

        if(idx == arr.length-1) {
            return true;
        }

        if(!checkIfIncreasing(arr, idx+1)) {
            return false;
        }

        return arr[idx] < arr[idx + 1];
    }

    public static void main(String args[]) {
        int arr1[] = {1, 2, 3, 4, 5};
        int arr2[] = {1, 6, 3, 4, 5};

        if(checkIfIncreasing(arr2, 0)) {
            System.out.println("Strictly Increasing");
        }else {
            System.out.println("NOT Strictly Increasing");
        }
    }
}

```

Q5. Move all 'x' to the end of the string. - $O(n)$

```

public class Recursion2 {
    //to add all 'x' to the end of the string
    public static String addX(int count) {
        String newStr = "x";
        for(int i=1;i<count; i++) {
            newStr += 'x';
        }
        return newStr;
    }

    public static String moveAllX(String str, int idx, int count) {
        if(idx == str.length()) {
            return addX(count);
        }

        if(str.charAt(idx) == 'x') {

```

```

        return moveAllX(str, idx+1, count+1);
    } else {
        String nextStr = moveAllX(str, idx+1, count);
        return str.charAt(idx) + nextStr;
    }
}

public static void main(String args[]) {
    String str = "abcdefxghxixjxxxk";
    int count = 0;

    String newStr = moveAllX(str, 0, count);
    System.out.println(newStr);
}
}

```

Q6. Remove duplicates in a string.

```

public class Recursion2 {
    public static String removeDuplicates(String str, int idx, boolean present[]) {
        if(idx == str.length()) {
            return "";
        }

        char curr = str.charAt(idx);
        if(present[curr-'a']) {
            return removeDuplicates(str, idx+1, present);
        } else {
            present[curr-'a'] = true;
            return curr + removeDuplicates(str, idx+1, present);
        }
    }

    public static void main(String args[]) {
        String str = "abcadbcefgghabi";
        boolean present[] = new boolean[str.length()];
        System.out.println(removeDuplicates(str, 0, present));
    }
}

```

Q7. Print all the subsequences of a string.

```
public class Recursion2 {
    public static void printSubseq(String str, int idx, String res) {
        if(idx == str.length()) {
            System.out.println(res);
            return;
        }

        //choose
        printSubseq(str, idx+1, res+str.charAt(idx));

        //don't choose
        printSubseq(str, idx+1, res);
    }
    public static void main(String args[]) {
        String str1 = "abc";
        String str2 = "aaa";

        printSubseq(str1, 0, "");
    }
}
```

Time complexity - $O(2^n)$

Q8. Print all unique subsequences of a string.

```
import java.util.HashSet;

public class Recursion2 {
    public static void printSubseq(String str, int idx, String res, HashSet<String>
allSubseq) {
        if(idx == str.length()) {
            if(allSubseq.contains(res)) {
                return;
            }
            allSubseq.add(res);
            System.out.println(res);
            return;
        }
    }
```

```

        //choose
        printSubseq(str, idx+1, res+str.charAt(idx), allSubseq);

        //don't choose
        printSubseq(str, idx+1, res, allSubseq);
    }
    public static void main(String args[]) {
        String str1 = "abc";
        String str2 = "aaa";
        HashSet<String> allSubseq = new HashSet<>();
        printSubseq(str2, 0, "", allSubseq);
    }
}

```

Q9. Print keypad combination

```

( 0 -> .;
  1 -> abc
  2 -> def
  3 -> ghi
  4 -> jkl
  5 -> mno
  6 -> pqrs
  7 -> tu
  8 -> vwx
  9 -> yz
)

```

```

import java.util.HashSet;

public class Recursion2 {
    public static String keypad[] = {".", "abc", "def", "ghi", "jkl", "mno", "pqrs",
    "tu", "vwx", "yz"};

    public static void printKeypadCombination(String number, int idx, String res) {
        if(idx == number.length()) {
            System.out.println(res);
            return;
        }
    }
}

```

```
        for(int i=0; i<keypad[number.charAt(idx)-'0'].length(); i++) {  
            char curr = keypad[number.charAt(idx)-'0'].charAt(i);  
            printKeypadCombination(number, idx+1, res+curr);  
        }  
    }  
    public static void main(String args[]) {  
        String number = "23";  
        printKeypadCombination(number, 0, "");  
    }  
}
```