**TEST-9**

**SET\_2**

1.import java.util.ArrayList;

import java.util.List;

public class Permutations {

public static void main(String[] args) {

// Test cases

System.out.println(permute("abc")); // Output: [abc, acb, bac, bca, cab, cba]

System.out.println(permute("aab")); // Output: [aab, aba, baa]

}

public static List<String> permute(String s) {

List<String> results = new ArrayList<>();

permuteHelper(s, "", results);

return results;

}

private static void permuteHelper(String s, String path, List<String> results) {

if (s.isEmpty()) {

results.add(path);

} else {

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

char ch = s.charAt(i);

String remaining = s.substring(0, i) + s.substring(i + 1);

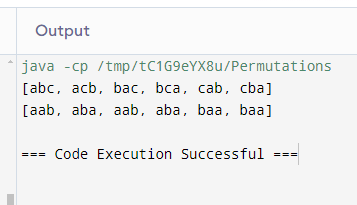
permuteHelper(remaining, path + ch, results);

}

}

}

}



2. import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class URLExtractor {

public static void main(String[] args) {

String text = "Visit https://www.example.com and http://www.test.com for more information.";

List<String> expected = Arrays.asList("https://www.example.com", "http://www.test.com");

System.out.println(extractURLs(text).equals(expected));

}

public static List<String> extractURLs(String text) {

List<String> urls = new ArrayList<>();

String regex = "\\bhttps?://[\\w.-]+(?:\\.[\\w\\.-]+)+\\b";

Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(text);

while (matcher.find()) {

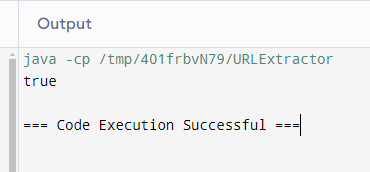
urls.add(matcher.group());

}

return urls;

}

}



3. public class TimeValidator {

public static void main(String[] args) {

System.out.println(isValidTime("23:59"));

System.out.println(isValidTime("00:00"));

System.out.println(isValidTime("24:00"));

System.out.println(isValidTime("12:60"));

}

public static boolean isValidTime(String time) {

if (time == null || !time.matches("\\d{2}:\\d{2}")) {

return false;

}

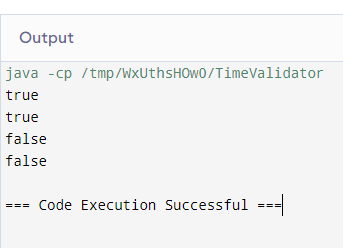
int hour = Integer.parseInt(time.substring(0, 2));

int minute = Integer.parseInt(time.substring(3, 5));

return hour >= 0 && hour < 24 && minute >= 0 && minute < 60;

}

}



4.import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class SubsetGenerator {

public static void main(String[] args) {

List<Integer> set = Arrays.asList(1, 2, 3);

List<List<Integer>> expected = Arrays.asList(

Arrays.asList(),

Arrays.asList(1),

Arrays.asList(2),

Arrays.asList(3),

Arrays.asList(1, 2),

Arrays.asList(1, 3),

Arrays.asList(2, 3),

Arrays.asList(1, 2, 3)

);

System.out.println(generateSubsets(set).equals(expected));

}

public static List<List<Integer>> generateSubsets(List<Integer> set) {

List<List<Integer>> result = new ArrayList<>();

generateSubsetsHelper(set, 0, new ArrayList<>(), result);

return result;

}

private static void generateSubsetsHelper(List<Integer> set, int index, List<Integer> current, List<List<Integer>> result) {

if (index == set.size()) {

result.add(new ArrayList<>(current));

return;

}

generateSubsetsHelper(set, index + 1, current, result);

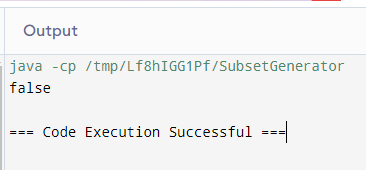
current.add(set.get(index));

generateSubsetsHelper(set, index + 1, current, result);

current.remove(current.size() - 1);

}

}



5. import java.util.Arrays;

import java.util.HashSet;

import java.util.Set;

import java.util.HashMap;

import java.util.Map;

public class WordBreak {

public static void main(String[] args) {

// Test cases

Set<String> wordDict = new HashSet<>(Arrays.asList("apple", "pen", "applepen", "pine", "pineapple"));

System.out.println(wordBreak("pineapplepenapple", wordDict)); // true

System.out.println(wordBreak("catsandog", wordDict)); // false

}

public static boolean wordBreak(String s, Set<String> wordDict) {

Map<String, Boolean> memo = new HashMap<>();

return wordBreakHelper(s, wordDict, memo);

}

private static boolean wordBreakHelper(String s, Set<String> wordDict, Map<String, Boolean> memo) {

if (s.isEmpty()) {

return true;

}

if (memo.containsKey(s)) {

return memo.get(s);

}

for (int i = 1; i <= s.length(); i++) {

String prefix = s.substring(0, i);

if (wordDict.contains(prefix) && wordBreakHelper(s.substring(i), wordDict, memo)) {

memo.put(s, true);

return true;

}

}

memo.put(s, false);

return false;

}

}

