1. 

**package** pretest.q1;

**import** java.util.LinkedHashSet;

**import** java.util.Set;

**public** **class** Solution {

**public** Set<String> findWords(Set<String> wordSet){

Set<String> set = **new** LinkedHashSet<>();

**for** (String word : wordSet) {

**if**(isOneLine4KeyBoard(word)){

set.add(word);

}

}

**return** set;

}

**private** **boolean** isOneLine4KeyBoard(String word){

**boolean** flag = **true**;

String wordUC = word.toUpperCase();

**char** first = wordUC.charAt(0);

**if**("QWERTYUIOP".indexOf(first)>-1){

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

**if** ("QWERTYUIOP".indexOf(wordUC.charAt(i))==-1) {

flag = **false**;

}

}

}

**if**("ASDFGHJKL".indexOf(first)>-1){

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

**if** ("ASDFGHJKL".indexOf(wordUC.charAt(i))==-1) {

flag = **false**;

}

}

}

**if**("ZXCVBNM".indexOf(first)>-1){

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

**if** ("ZXCVBNM".indexOf(wordUC.charAt(i))==-1) {

flag = **false**;

}

}

}

**return** flag;

}

**public** **static** **void** main(String[] args) {

Solution solution = **new** Solution();

Set<String> set = **new** LinkedHashSet<>();

set.add("life");

set.add("require");

set.add("option");

set.add("hall");

System.***out***.println(solution.findWords(set));

}

}

1. 

**package** pretest.q2;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Solution {

**private** **class** Node{

**public** Node left;

**public** Node right;

**public** **int** value;

@Override

**public** String toString() {

**return** "Node:"+ value;

}

}

**public** List<Node> findPath(Node rootNode,**int** sum){

ArrayList<ArrayList<Node>> paths = **new** ArrayList<>();

findPaths(rootNode, 15, paths);

**return** paths.get(0);

}

**private** ArrayList<Node> nodes = **new** ArrayList<>();

**private** **int** curSum = 0;

**private** **void** findPaths(Node rootNode,**int** sum,ArrayList<ArrayList<Node>> paths){

**if**(rootNode == **null**){

**return**;

}

curSum += rootNode.value;

nodes.add(rootNode);

**if** (sum == curSum && rootNode.left == **null** && rootNode.right == **null**){

paths.add((ArrayList<Node>)nodes.clone());

}

**if** (rootNode.left != **null**) {

findPaths(rootNode.left, sum , paths);

}

**if** (rootNode.right != **null**) {

findPaths(rootNode.right, sum , paths);

}

curSum -= rootNode.value;

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

}

**public** **static** **void** main(String[] args) {

Solution sol = **new** Solution();

Node rootNode = sol.**new** Node();

rootNode.value = 5;

rootNode.left = sol.**new** Node();

rootNode.left.value = -2;

rootNode.left.left = **null**;

rootNode.left.right = sol.**new** Node();

rootNode.left.right.value = 1;

rootNode.left.right.left = **null**;

rootNode.left.right.right = sol.**new** Node();

rootNode.left.right.right.value = 10;

rootNode.left.right.right.left = **null**;

rootNode.left.right.right.right = **null**;

rootNode.right = sol.**new** Node();

rootNode.right.value = -3;

rootNode.right.left = sol.**new** Node();

rootNode.right.left.value = 5;

rootNode.right.left.left = sol.**new** Node();

rootNode.right.left.left.value = 8;

rootNode.right.left.left.left = **null**;

rootNode.right.left.left.right = **null**;

rootNode.right.left.right = **null**;

rootNode.right.right = sol.**new** Node();

rootNode.right.right.value = 9;

rootNode.right.right.left = sol.**new** Node();

rootNode.right.right.left.value = 4;

rootNode.right.right.right = sol.**new** Node();

rootNode.right.right.right.value = 4;

rootNode.right.right.left.left = **null**;

rootNode.right.right.left.right = **null**;

rootNode.right.right.right.left = **null**;

rootNode.right.right.right.right = **null**;

System.***out***.println(sol.findPath(rootNode,15));

}

}