**package** nyc.c4q;  
  
**import** java.util.\*;  
  
**public class** FindTheKey {  
 **private static final int *MAP\_SIZE\_RANGE*** = 3;  
 **private static final int *SIZE\_ONE*** = 5;  
 **private static final int *SIZE\_TWO*** = 2;  
 **private static final int *SIZE\_OF\_ALPHABET*** = 26;  
 **private static final int *ASCII\_CAPITAL\_A*** = 65;  
  
 **private final** Map<String,Integer> **keyValues**;  
 **private final** Random **random** = **new** Random();  
  
 **public** FindTheKey(){  
 **keyValues** = **new** HashMap<>();  
 **int** mapSize = **random**.nextInt(***MAP\_SIZE\_RANGE***) + 1;  
 **for** (**int** index = 0; index < mapSize; index++) {  
 **keyValues**.put(randomString(), index);  
 }  
 }  
  
 **private** String randomString() {  
 **int** tempA = **random**.nextInt(***SIZE\_ONE***) + ***SIZE\_TWO***;  
 String value = **""**;  
 **for** (**int** index = 0; index < tempA; index++){  
 value += (**char**)(**random**.nextInt(***SIZE\_OF\_ALPHABET***) + ***ASCII\_CAPITAL\_A)***;  
 }  
 **return** value;  
 }

**public** List<String> listOfKeys(){  
 Set<String> keys = **keyValues**.keySet();  
 ArrayList<String> result = **new** ArrayList<>();  
 **for** (String s : keys){  
 result.add(s);  
 }  
 **return** result;  
}

**public void** printAllKeys() {  
 // TODO  
 }  
}

**package** nyc.c4q;  
  
*/\*\*  
 \* Write a class that checks if a letter is a palidrome  
 \*/***public class** Palindrome {  
 **private final** String **input**;  
 **private boolean isPalindrome**;  
 **private boolean finishedCalculations**;  
  
 **public** Palindrome(String testString) {  
 **input** = testString;  
 **isPalindrome** = **false**;  
 **finishedCalculations** = **false**;  
 }  
  
 **public boolean** isPalindrome() {  
 **if** (!**finishedCalculations**) {  
 **isPalindrome** = doPalindromeCheck();  
 **finishedCalculations** = **true**;  
 }  
 **return isPalindrome**;  
 }  
  
 **private boolean** doPalindromeCheck() {  
 **// TODO**  
 }  
}

**package** nyc.c4q;  
  
**import** java.util.ArrayList;  
**import** java.util.Stack;  
  
**public class** StackingIt {  
 **private** Stack<Integer> **stack** = **new** Stack<>();  
  
 **public** StackingIt(String input){  
 **for** (**char** c : input.toCharArray()){  
 **stack**.push((**int**) c);  
 }  
 }  
  
 **public void** popThenAdd(){  
 **while** (**stack**.size() > 2){  
 **int** value1 = **stack**.pop();  
 **int** value2 = **stack**.pop();  
 **stack**.push(value1+value2);  
 }  
 }  
  
 **public void** popThenAppend(){  
 **while** (**stack**.size() != 1){  
 **int** value1 = **stack**.pop();  
 **int** value2 = **stack**.pop();  
 String result = value1 + **""** + value2;  
 **stack**.push(Integer.*valueOf*(result));  
 }  
 }  
  
 **public void** printStack(){  
 ArrayList<Integer> output = **new** ArrayList<>();  
 **while** (!**stack**.isEmpty()) {  
 output.add(**stack**.pop());  
 }  
 **for** (**int** index = output.size()-1; index >= 0; index--){  
 System.***out***.println(output.get(index));  
 }  
 }  
  
  
}

**package** nyc.c4q;  
  
**import** java.util.LinkedList;  
**import** java.util.List;  
  
**public class** TravelTheList {  
 **private** LinkedList<String> **stringList** = **new** LinkedList<>();  
 **public** TravelTheList(){  
 **stringList**.add(**"Hello"**);  
 **stringList**.add(**"Middle"**);  
 **stringList**.add(**"Goodbye"**);  
 **stringList** = expandList(0);  
 **stringList** = expandList(**stringList**.size() -2);  
 **stringList** = expandList(**stringList**.size() -1);  
 }  
  
 **public** TravelTheList(List<String> input){  
 **for** (String s : input) {  
 **stringList**.add(s);  
 }  
 **for** (**int** index = **stringList**.size(); index >= 1; index--){  
 **stringList** = expandList(**stringList**.size() - index);  
 }  
 }  
  
 **private** LinkedList<String> expandList(**int** startPoint) {  
 LinkedList<String> result = **new** LinkedList<>();  
 **for** (**int** index = 0; index < startPoint; index++){  
 result.add(**stringList**.get(index));  
 }  
 String stringToExpand = **stringList**.get(startPoint);  
 result.add(stringToExpand);  
 **for** (**int** index = 0; index < stringToExpand.length(); index++){  
 result.add(**""** + stringToExpand.charAt(index));  
 }  
 **for** (**int** index = startPoint+1; index < **stringList**.size(); index++){  
 result.add(**stringList**.get(index));  
 }  
 **return** result;  
 }  
  
 **public void** printStringList(){  
 **for** (String s : **stringList**){  
 System.***out***.println(s);  
 }  
 }  
}



**Interface List<E>**

E get(int index)

Returns the element at the specified position in this list.

boolean add(E e)

Appends the specified element to the end of this list.

**Interface Map<K,V>**

boolean isEmpty()

Returns true if this map contains no key-value mappings.

Set<K> keySet()

Returns a Set view of the keys contained in this map.

V put(K key, V value)

Associates the specified value with the specified key in this map (optional operation).

V get(Object key)

Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.

## Class Random

An instance of this class is used to generate a stream of pseudorandom numbers.

int nextInt(int n)

Returns a pseudorandom, uniformly distributed int value between 0 (inclusive) and the specified value (exclusive), drawn from this random number generator's sequence.

## Class Stack<E>

E pop()

Removes the object at the top of this stack and returns that object as the value of this function.

E push(E item)

Pushes an item onto the top of this stack.

Question 1:

Finish the function FindTheKey.printAllKeys();

Write code that will print all of the key value pairs in the keyValues map like so:

Key1 - Value1

Key2 – Value2

Question 2:

Finish the function Palidrome.doPalidromeCheck().

Write code to check whether the input field is a palindrome.

A string is a palindrome if it remains unchanged when reversed.

Question 3:

What is the output to the terminal of the following function:

StackingIt s = **new** StackingIt(**"AAA"**);  
s.popThenAdd();  
s.popThenAppend();  
s.printStack();

Question 4:

What is the output to the terminal of the following function:

List<String> in = Arrays.*asList*(**"cat"**, **"dog"**, **"go"**);  
**new** TravelTheList(test).printStringList();

Question 5:

The following questions refer to class FindTheKey:

1. What is the smallest size keyValues can be?
2. What is the largest size keyValues can be?
3. In the randomString() function, what does tempA variable determine?
4. If keyValues was of type Map<Monkeys,Fish>, what type would keyValues.keyset() return?

Question 6:

How many times is doPalidromeCheck() called in the following function?

Palidrome p = new Palidrome(“start"  
p.isPalidrome();  
p.isPalidrome();  
p = **new** Palidrome(**"help"**);  
p = **new** Palidrome(**"done"**);  
p.isPalidrome();