

**package** MorganStanley;

**import** java.util.Arrays;

**public** **class** MSTest {

// public static int lengthOfLIS(int[] nums) {

// int memo[][] = new int[nums.length + 1][nums.length];

// for (int[] l : memo) {

// Arrays.fill(l, -1);

// }

// return lengthofLIS(nums, -1, 0, memo);

// }

//

// public static int lengthofLIS(int[] nums, int previndex, int curpos, int[][] memo) {

// if (curpos == nums.length) {

// return 0;

// }

// if (memo[previndex + 1][curpos] >= 0) {

// return memo[previndex + 1][curpos];

// }

// int taken = 0;

// if (previndex < 0 || nums[curpos] > nums[previndex]) {

// taken = 1 + lengthofLIS(nums, curpos, curpos + 1, memo);

// }

//

// int nottaken = lengthofLIS(nums, previndex, curpos + 1, memo);

// memo[previndex + 1][curpos] = Math.max(taken, nottaken);

// return memo[previndex + 1][curpos];

// }

**public** **static** **int** lengthOfLIS(**int**[] nums) {

**if** (nums.length == 0) {

**return** 0;

}

**int**[] dp = **new** **int**[nums.length];

dp[0] = 1;

**int** maxans = 1;

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

**int** maxval = 0;

**for** (**int** j = 0; j < i; j++) {

**if** (nums[i] > nums[j]) {

maxval = Math.*max*(maxval, dp[j]);

}

}

dp[i] = maxval + 1;

maxans = Math.*max*(maxans, dp[i]);

}

**return** maxans;

}

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

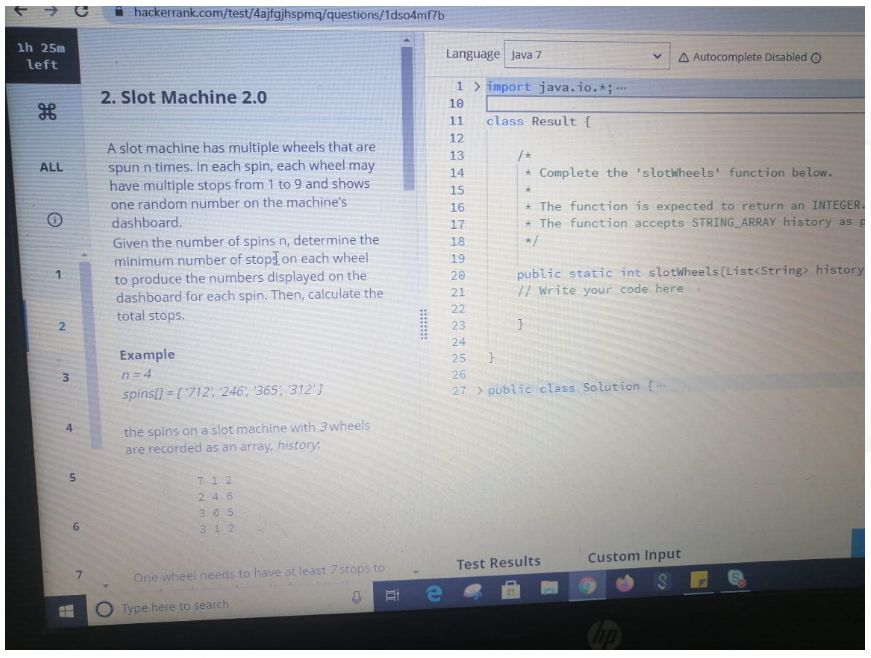
// **TODO** Auto-generated method stub

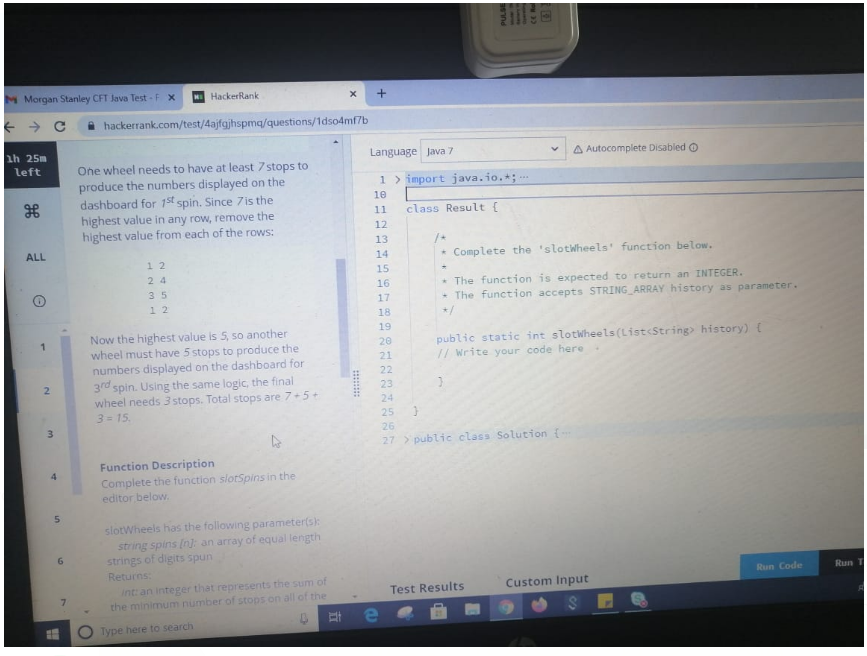
**int** s[] = { 1, 4, 3 };

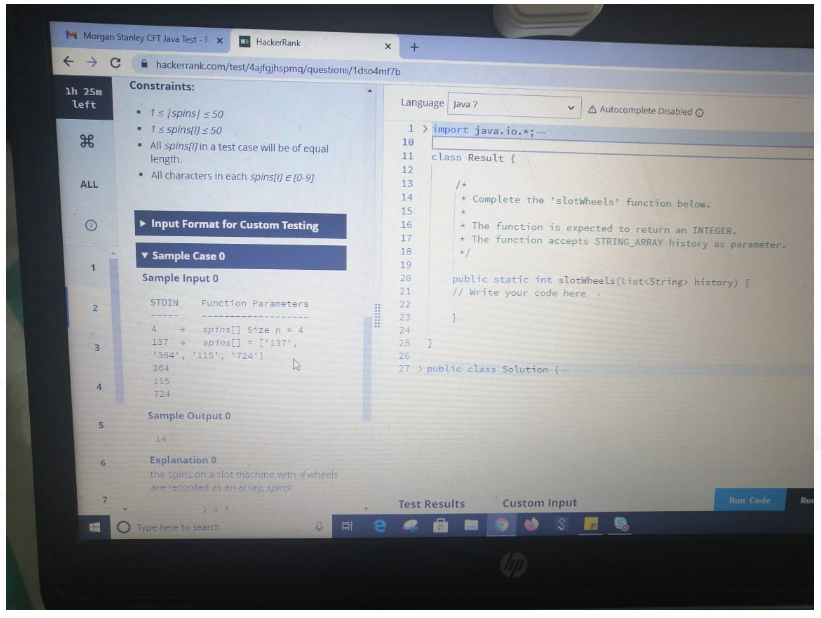
System.***out***.print(*lengthOfLIS*(s));

}

}







**package** MorganStanley;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.List;

**public** **class** SlotMachine {

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

String spins[] = **new** String[]{"712", "246", "365", "312"};

System.***out***.println(*solve*(spins, 0));

}

**private** **static** **int** solve(String[] spins, **int** sum) {

ArrayList<Integer> max\_arr = **new** ArrayList<>();

**if** (spins.length > 0 && spins[0].length() > 0) {

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

**int** maxVal = Integer.***MIN\_VALUE***;

**for** (String c : spins[i].split("")) {

**int** digit = Integer.*parseInt*(c);

maxVal = Math.*max*(digit, maxVal);

}

max\_arr.add(maxVal);

spins[i] = spins[i].replace(Integer.*toString*(maxVal), "");

}

sum += Collections.*max*(max\_arr);

**return** *solve*(spins, sum);

}

**return** sum;

}

/\*\*\*\*\*\*\* Solution two \*\*\*\*\*\*/

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

// List<String> ip = new ArrayList<>();

// ip.add("712");

// ip.add("246");

// ip.add("365");

// ip.add("312");

//

// int op = solve(ip);

// System.out.println(op);

// }

// private static int solve(List<String> ip) {

// MyKlass myKlass = new MyKlass();

// myKlass.currLen = ip.get(0).length();

// solveByRec(myKlass, ip);

// return myKlass.sum;

// }

//

// private static void solveByRec(MyKlass myKlass, List<String> ip) {

// // base cond

// if (ip.size() > 0 && ip.get(0).length() == 0)

// return;

//

// // solution

// int i = 0;

// int thisIterMax = Integer.MIN\_VALUE;

// for (String a : ip) {

// String t = a;

// int currMax = getMaxFromString(t);

// String newString = getNewStringByRemovingMax(t, currMax);

// ip.set(i, newString);

// thisIterMax = Math.max(thisIterMax, currMax);

// i++;

// }

// myKlass.sum += thisIterMax;

// solveByRec(myKlass, ip);

// }

//

// private static String getNewStringByRemovingMax(String t, int currMax) {

// StringBuilder newStr = new StringBuilder();

// boolean flag = true;

// for (char c : t.toCharArray()) {

// if (flag && ((int) (c - '0')) == currMax) {

// // don't add

// flag = false;

// } else {

// newStr.append(c);

// }

// }

// return newStr.toString();

// }

//

// private static int getMaxFromString(String t) {

// char[] arr = t.toCharArray();

// int max = Integer.MIN\_VALUE;

// for (char c : arr) {

// max = Math.max(max, (int) (c - '0'));

// }

// return max;

// }

//

// static class MyKlass {

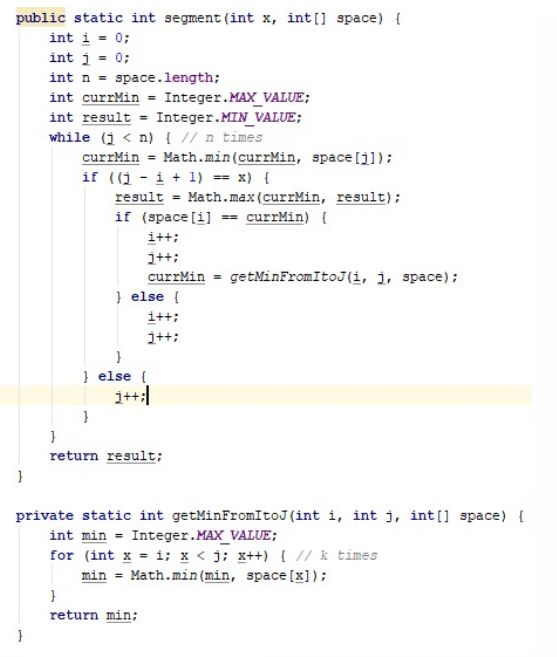
// int sum = 0;

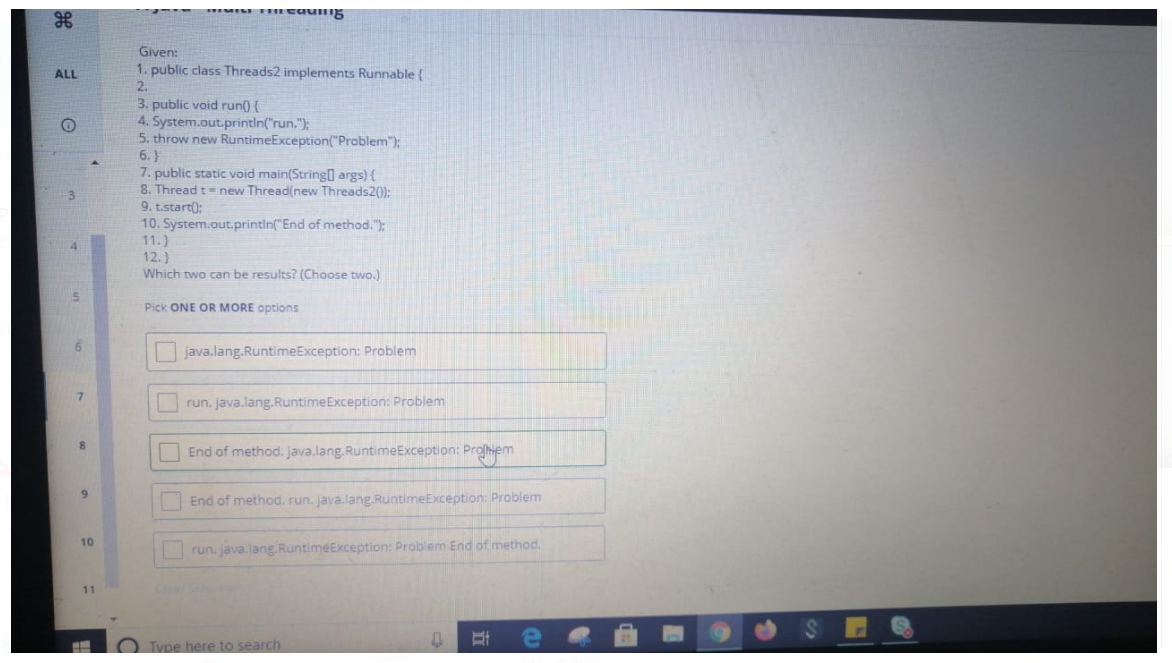
// int currLen;

// }

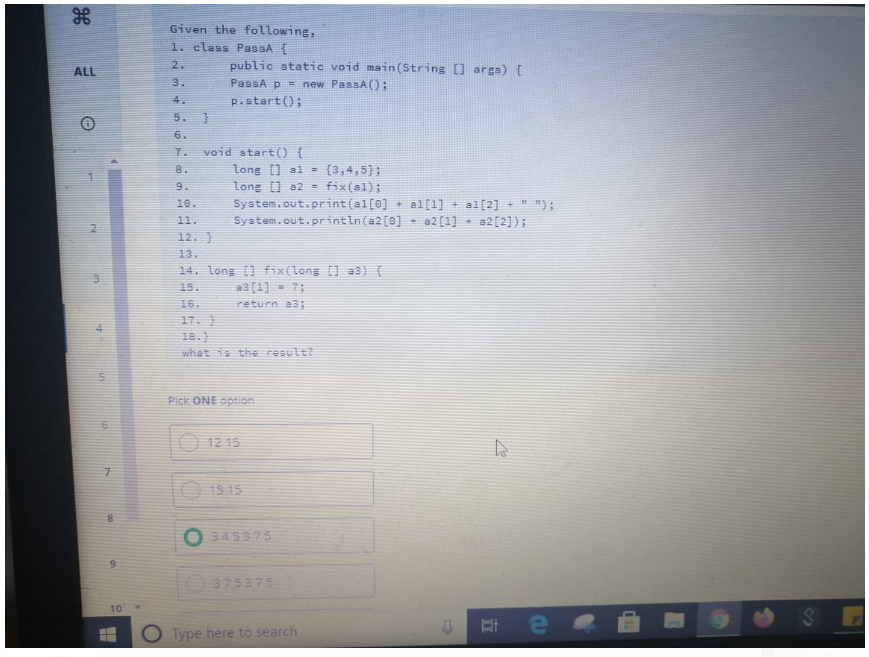
}

\*\*\*\*\*\*\*\*\*\*\*\*\* Disk space analysis \*\*\*\*\*\*\*\*\*\*

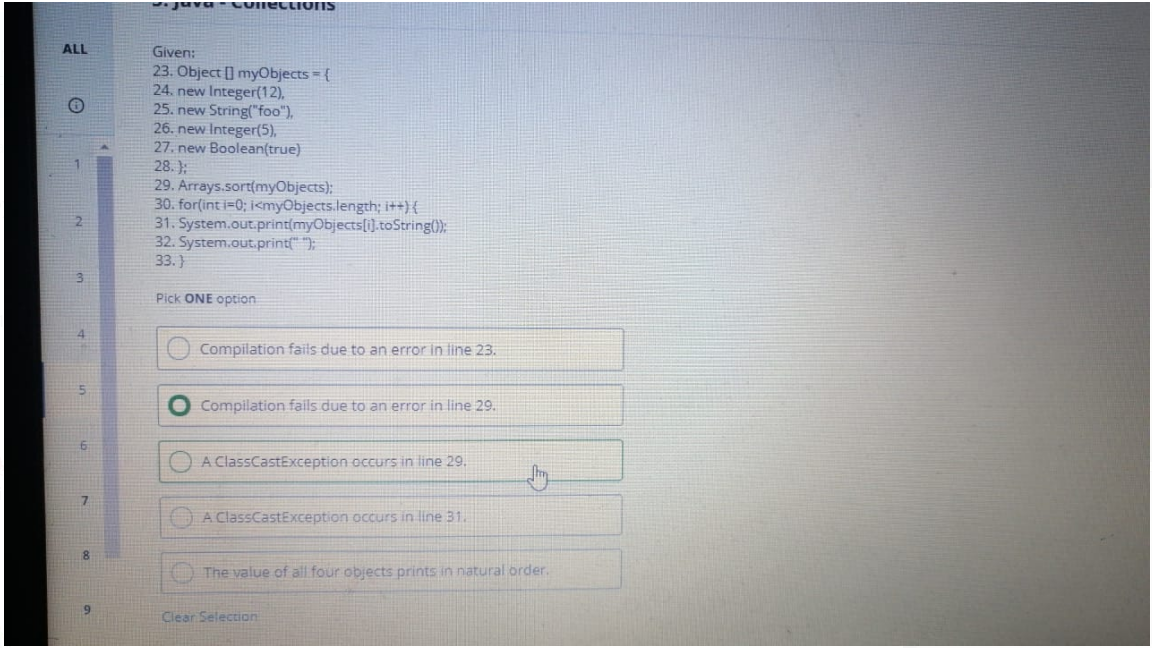




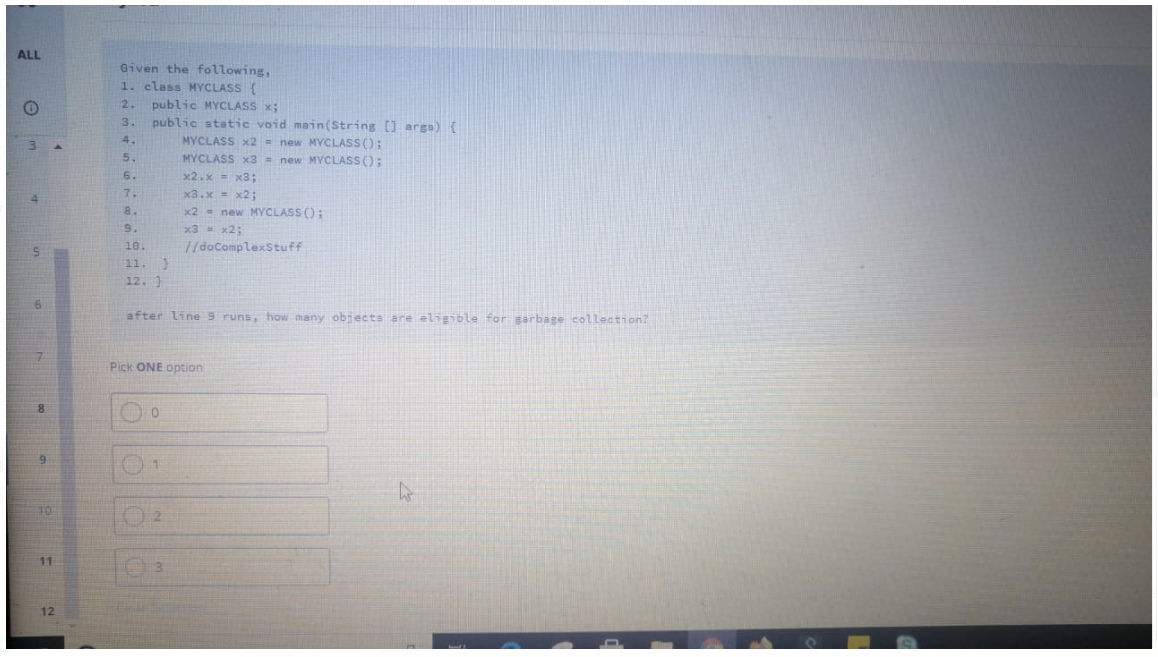
Answer : 2 and 5



Answer : 15 15



Answer : 3



Answer : 0

Question :

**package** MorganStanley;

**public** **class** SampleTest {

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

// **TODO** Auto-generated method stub

Animal a1=**new** Animal("Lion");

Animal a2=**new** Animal("Crocodile");

System.***out***.println("Before swap: a1:" + a1 + "; a2:"+ a2);

*swap*(a1,a2);

System.***out***.println("After swap: a1:" + a1 + "; a2:"+ a2);

}

**public** **static** **void** swap(Animal an1,Animal an2) {

Animal tmp= **new** Animal("");

tmp=an1;

an1=an2;

an2=tmp;

}

}

**class** Animal{

String name;

**public** Animal(String name) {

**this**.name=name;

}

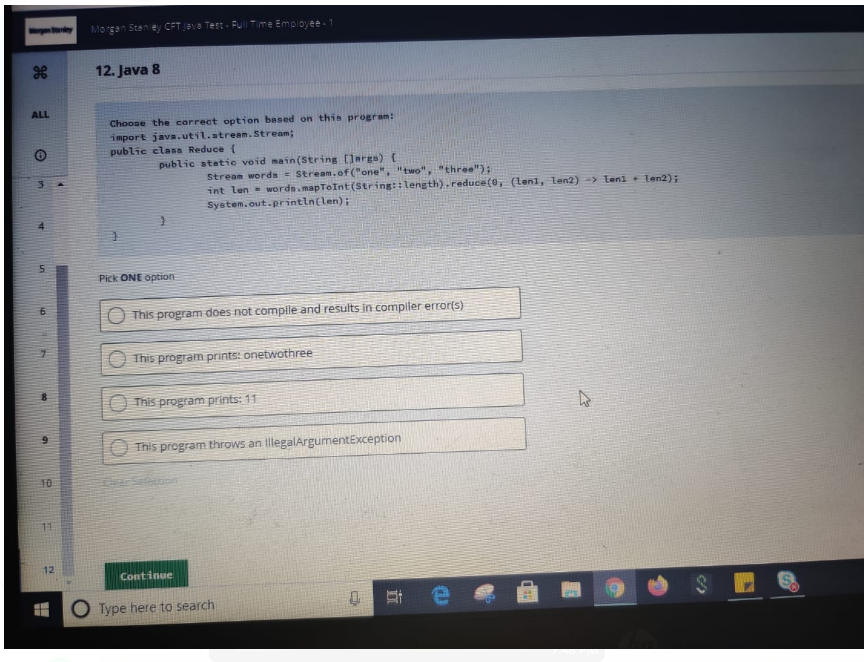
**public** String toString() {

**return** name;

}

}

Answer : Before swap: a1:Lion; a2:Crocodile After swap: a1:Lion; a2:Crocodile



Answer : 1