```
1
2  # Stock Max Difference
3
4  # Given an array of ints, determine the max profit from buying
5  # and selling once each. You must buy before you sell. if the array continuously
6  # decreases, return -1.
7
8  def maxDifference( a):
9      maxDiff = -1
10     low = a[0]
11
12     for i in a[1:]:
13         if i - low > maxDiff: maxDiff = i - low
14         if i < low: low = i
15
16     return maxDiff
17
18 # Time Complexity: Primality
19
20 # If possible, try to come up with an  primality algorithm, or see what
21 # sort of optimizations you can come up with for an  algorithm.
22 # https://www.hackerrank.com/challenges/ctci-big-o/copy-from/30793312
23
24 import math
25
26 def isPrime(n):
27     if n == 2:
28         return True
29     elif n == 1 or (n & 1) == 0:
30         return False
31
32     for i in range(2, math.ceil(math.sqrt(n)) + 1):
33         if (n % i) == 0:
34             return False
35
36     return True
37
38 p = int(input())
39 for i in range(0, p):
40     x = int(input())
41
42     s = "Prime" if (isPrime(x)) else "Not prime"
43     print(s);
44
45
46
47
```

```
48 # DFS: Connected Cell in a Grid
49
50 # Given an n x m matrix, find and print the number of cells in the largest region in the
51 # matrix. Note that
52 # there may be more than one region in the matrix.
53 # https://www.hackerrank.com/challenges/ctci-connected-cell-in-a-grid
54
55 def get_biggest_region(grid):
56   def bfs(grid, x, y):
57     queue = [[x,y]]
58     start, end = 0, 1
59
60     while (start < end):
61       for pos in xrange(start, end):
62         curPos = queue[pos]
63         for i in xrange(-1,2):
64           for j in xrange(-1, 2):
65             aX, aY = curPos[0] + i, curPos[1] + j
66
67             if [aX, aY] not in queue and -1 < aY < len(grid) \
68             and -1 < aX < len(grid[0]) and grid[aY][aX]:
69               queue.append([aX, aY])
70               grid[aY][aX] = 0
71       start = end
72       end = len(queue)
73
74     return len(queue)
75
76   maxReg = 0
77
78   for y in xrange(len(grid)):
79     for x in xrange(len(grid[0])):
80       if grid[y][x]:
81         maxReg = max(maxReg, bfs(grid, x, y))
82
83   return maxReg
84
85 n = int(raw_input().strip())
86 m = int(raw_input().strip())
87 grid = []
88 for grid_i in xrange(n):
89     grid_temp = map(int, raw_input().strip().split(' '))
90     grid.append(grid_temp)
91 print get_biggest_region(grid)
92
93
94
```

```
95  # Recursion: Davis' Staircase
96
97  # Davis has  staircases in his house and he likes to climb each staircase
98  # 1, 2, 3 steps at a time. Given the respective heights for
99  # each of the  staircases in his house, find and print the number of ways
100 # he can climb each staircase on a new line.
101 # https://www.hackerrank.com/challenges/ctci-recursive-staircase
102
103 recordRec = [1,1,2] + [-1]*34
104 record = [1,1,2]
105
106 def countRec(x):
107   if len(recordRec) > x and record[x] != -1:
108     return recordRec[x]
109   else:
110     recordRec[x] = countRec(x - 1) + countRec(x - 2) + countRec(x - 3)
111     return recordRec[x]
112
113 def count(x):
114   if len(record) > x:
115     return record[x]
116   else:
117     for i in xrange(len(record)-1, x+1):
118       record.append(record[-1] + record[-2] + record[-3])
119   return record[x]
120
121 s = int(raw_input().strip())
122 for a0 in xrange(s):
123     n = int(raw_input().strip())
124     print countRec(n)
125
126
127 # Time Complexity: Primality
128
129 # If possible, try to come up with an  primality algorithm, or see what
130 # sort of optimizations you can come up with for an  algorithm.
131 # https://www.hackerrank.com/challenges/ctci-big-o/copy-from/30793312
132
133 import math
134
135
136
137
138
139
140
141
```

```
142 def isPrime(n):
143     if n == 2:
144         return True
145     elif n == 1 or (n & 1) == 0:
146         return False
147 
148     for i in range(2, math.ceil(math.sqrt(n)) + 1):
149         if (n % i) == 0:
150             return False
151 
152     return True
153 
154 p = int(input())
155 for i in range(0, p):
156     x = int(input())
157 
158     s = "Prime" if (isPrime(x)) else "Not prime"
159     print(s);
160 # Sorting Comparator
161 
162 class Player:
163     def __init__(self, name, score):
164         self.name = name
165         self.score = score
166 
167     def __repr__(self):
168         return 'player(name=%s, score=%s)' % (self.name, string(self.score))
169 
170 
171     def comparator(a, b):
172         if a.score == b.score:
173             if a.name < b.name:
174                 return -1
175             else:
176                 return 1
177         else:
178             if  a.score < b.score:
179                 return 1
180             else:
181                 return -1
182 
183 
184 
185 
186 
187 
188 
```

```
189 # Binary Search: Ice Cream Parlor
190
191 def pick2(m, costs):
192     d = {}
193     for i in xrange(len(costs)):
194         #print str(costs[i])
195         #print d
196         if costs[i] in d:
197             print (str(d[costs[i]] + 1) + " " + str(i+1))
198         else:
199             d[m-costs[i]] = i
200
201
202 t = int(raw_input().strip())
203 for a0 in xrange(t):
204     m = int(raw_input().strip())
205     n = int(raw_input().strip())
206     a = map(int, raw_input().strip().split(' '))
207     pick2(m, a)
208 # Binary Search: Ice Cream Parlor
209
210 def pick2(m, costs):
211     d = {}
212     for i in xrange(len(costs)):
213         #print str(costs[i])
214         #print d
215         if costs[i] in d:
216             print (str(d[costs[i]] + 1) + " " + str(i+1))
217         else:
218             d[m-costs[i]] = i
219
220
221 t = int(raw_input().strip())
222 for a0 in xrange(t):
223     m = int(raw_input().strip())
224     n = int(raw_input().strip())
225     a = map(int, raw_input().strip().split(' '))
226     pick2(m, a)
227
228 # HackerRank: CTCI
229 # Tree: is this a BST
230
231
232
233
234
235
```

```
236 def check_binary_search_tree_(root):
237     arr = []
238     count = 0
239     arr = inorderTraversal(root, arr)
240     if ((sorted(arr)) == arr) and (len(set(arr)) == len(arr)):
241         return True
242     else:
243         return False
244 
245 def inorderTraversal(root, arr):
246     if root != None:
247         inorderTraversal(root.left, arr)
248         arr.append(root.data)
249         inorderTraversal(root.right, arr)
250     return arr
251 
252 
253 
254 # given an array of integers and a number k, find the number of pairs
255 # in the array such that x + k = y
256 
257 def  kDifference(a, k):
258     dict = {}
259     count = 0
260     for x in a:
261         if x not in dict:
262             dict[x] = True
263     print dict
264     for x in a:
265         if (x+k) in dict:
266             count += 1
267         if ((x-k) > 0) and (x-k) in dict:
268             count += 1
269     return count / 2
270 
271 
272 # MakingAnagrams
273 
274 # https://www.hackerrank.com/challenges/ctci-making-anagrams/submissions/code/29308263
275 
276 # Given two strings,  and , that may or may not be of the same length,
277 # determine the minimum number of character deletions required to make  and
278 # anagrams. Any characters can be deleted from either of the strings
279 
280 
281 
282 
```

```
def number_needed(a, b):
    dup_count = 0
    alphabet = [0]*26;
    counter = update_counter(b, update_counter(a, alphabet, True), False)

    #print counter
    for i in counter:
        if i == 0:
            continue
        else:
            dup_count += abs(i)
    return dup_count



def update_counter(word, counter, add):
    ascii_values = [ord(letter)-97 for letter in word]
    if add:
        for val in ascii_values:
            counter[val] += 1
        return counter
    else:
        for val in ascii_values:
            counter[val] -=1
        return counter


a = raw_input().strip()
b = raw_input().strip()
print number_needed(a, b)
# Array Rotation

# https://www.hackerrank.com/challenges/ctci-array-left-rotation

# A left rotation operation on an array of size  shifts each of the array's
# elements  unit to the left.
# For example, if left rotations are performed on array ,
# then the array would become .

# Given an array of  integers and a number, , perform  left rotations on the array.
# Then print the updated array as a single line of space-separated integers.

import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
```

```

public class Solution {

    public static int[] arrayLeftRotation(int[] a, int n, int k) {
        int[] shifted = new int[a.length];
        for (int i=0; i<a.length; i++) {
            if (i-k < 0) {
                shifted[a.length - (k-i)] = a[i];
            } else {
                shifted[i-k] = a[i];
            }
        }
        return shifted;
    }

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int k = in.nextInt();
        int a[] = new int[n];
        for(int a_i=0; a_i < n; a_i++){
            a[a_i] = in.nextInt();
        }

        int[] output = new int[n];
        output = arrayLeftRotation(a, n, k);
        for(int i = 0; i < n; i++)
            System.out.print(output[i] + " ");

        System.out.println();

    }
}
# string compression

# https://www.careercup.com/question?id=7449675
# asked by Yelp and Amazon

# Compress a given string "aabbbccc" to "a2b3c3"
# constraint: inplace compression, no extra space to be used
# assumption : output size will not exceed input size.. ex input:"abb" ->
# "a1b2" buffer overflow.. such inputs will not be given.

# https://www.hackerrank.com/challenges/string-compression

given = "abcaaabbb"
output = "abca3b3"
```

```
377
378 def compressor(given):
379
380 # Brackets Question (Tony)
381
382 # Find all possible valid combination of brackets given a number of brackets possible
383 # e.g. 3 -> {}{}{}, {{{}}}, {}{{}}, {{}}{} -> 4
384
385 def brackets(p_arr, n):
386   if (sum(p_arr) == n):
387     print ''.join(['['*i + ']'*i for i in p_arr])
388     return;
389
390   for i in xrange(1, n - sum(p_arr)+1):
391     brackets(p_arr + [i],n)
392
393 brackets([],8)
394 # Fibonacci
395
396
397 ## Example 1: Using looping technique
398 def fib(n):
399  a,b = 1,1
400  for i in range(n-1):
401   a,b = b,a+b
402  return a
403 print fib(5)
404
405 ## Example 2: Using recursion
406 def fibR(n):
407  if n==1 or n==2:
408   return 1
409  return fib(n-1)+fib(n-2)
410 print fibR(5)
411
412
```