1. static int nextPerfectSquare(int n) {

int nextPerfectSquare = 1;

if (n < 0)

return 0;

else if (n == 0)

return 1;

for (int i = 0; i < n; i++)

if (i \* i > n)

{

nextPerfectSquare = i \* i;

break;

}

return nextPerfectSquare;

}

static int isPerfectSquare(int n) {

if (n == 0)

return 1;

if (n < 0)

return 0;

int square;

int number = 1;

do {

square = number \* number;

number++;

} while (square <= n);

return square;

}

2. static int nUpCount(int[] a, int n) {

int counter = 0;

int previousPartialSum = 0;

int partialSum = 0;

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

previousPartialSum = partialSum;

partialSum += a[i];

if (previousPartialSum <= n && partialSum > n)

counter ++;

}

return counter;

}

2. static int primeCount(int start, int end) {

int count = 0;

for(int j= start; j <= end; j++) {

if(isPrime(j))

count ++;

}

return count;

}

2. [From knight]

static int isMadhavArray(int[] a) {

boolean correctLength = false;

int n = 2;

double checkLength = 1;

while (checkLength <= a.length) {

checkLength = n \* (n + 1) / 2.0;

if (checkLength == a.length) {

correctLength = true;

break;

}

n++;

}

if (!correctLength) return 0;

int index = 1;

for (int i = 2; index < a.length; i++) {

int sum = 0;

for (int j = 1; j <= i; j++) {

sum += a[index];

index++;

}

if (sum != a[0]) return 0;

}

return 1;

}

}

4. static int inertial(int[] a) {

int oddCount = 0;

int maxValue = a[0];

boolean oddGreater = true;

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

if(a[i]%2 != 0)

oddCount ++;

if(a[i] > maxValue)

maxValue = a[i];

}

for(int k=0; k < a.length; k++) {

if(a[k]%2 != 0) {

for(int j=0; j < a.length; j++) {

if(a[j]%2 == 0 && a[j] != maxValue) {

if(a[k] < a[j])

oddGreater = false;

}

}

}

}

if(oddCount == 0) {

return 0;

}

if(maxValue %2 != 0)

return 0;

if(oddGreater)

return 1;

return 0;

}

5. static int countSquarePair(int[] a) {

int count = 0;

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

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

if (a[i] > 0 && a[j] > 0 && a[i] < a[j] ) {

if (isPerfectSquare(a[i] + a[j])) {

count++;

}

}

}

}

return count;

}

static int isPerfectSquare(int n) {

boolean isSquare = false;

if(n < 0)

return 0;

for(int i = 1; i <= n/2; i++) {

if(n%i == 0) {

int multiple = n/i;

if(multiple == i)

isSquare = true;

}

}

if(isSquare || n == 0)

return 1;

return 0;

}

5. static int findPorcupineNumber(int n)

{

int porcupine = n + 1;

while (true)

{

if (isPrime(porcupine))

{

int digit = porcupine % 10;

if (digit == 9)

{

int nextPrime = porcupine + 1;

while (true)

{

if (isPrime(nextPrime))

{

int nextDigit = nextPrime % 10;

if (nextDigit == 9)

{

return porcupine;

}

else

{

porcupine = nextPrime;

break;

}

}

nextPrime++;

}

}

}

porcupine++;

}

}

6. static int isGuthrieSequence(int[] a) {

if (a[a.length - 1] != 1)

return 0;

for (int i = 0; i < a.length - 1; i++) {

if (a[i] % 2 == 0) {

if ( (a[i] / 2)!= a[i + 1])

return 0;

}

else

{

if (((a[i] \* 3) + 1) != a[i + 1] )

return 0;

}

}

return 1;

}

6. static int stantonMeasure(int[] a) {

int n = 0;

int count =0;

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

if(a[i] == 1)

n++;

}

for(int j=0; j< a.length; j++) {

if(a[j] == n)

count++;

}

return count;

}

7. static int sumFactor(int[] a) {

int sum = 0;

int sumFactor = 0;

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

sum += a[i];

}

for(int j=0; j< a.length; j++) {

if(a[j] == sum)

sumFactor ++;

}

return sumFactor;

}

8. public class GuthrieIndex {

static int guthrieIndex(int n) {

int count = 0;

while(n>1) {

if(n%2 == 0) {

n /= 2;

count ++;

}

else if(n % 2 != 0) {

n \*= 3;

n += 1;

count ++;

}

}

return count;

}

public static void main(String[] args) {

System.out.println(guthrieIndex(3));

}

9. int repsEqual(int[] a, int n) {

int lengthArray = a.length;

int multipleOfTen = 1;

int arrayToNumber = 0;

for(int i= lengthArray-1; i >= 0; i--) {

arrayToNumber += (a[i]\*multipleOfTen);

multipleOfTen \*= 10;

}

if(arrayToNumber == n)

return 1;

return 0;

}

10. static int isCentered15(int[] a) {

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

int sum = 0;

for (int j = i; j < a.length; j++) {

sum += a[j];

if (sum == 15) {

if (i == a.length - j - 1)

return 1;

}

}

}

return 0;

}

}

10. boolean isPerfectNumber(int n) {

int sum = 0;

for(int i = 1; i < n; i++)

{

if(n%i == 0)

sum += i;

}

if(sum == n)

return true;

else

return false;

}

int henry(int i, int j) {

int count = 0;

int henrySum = 0;

for(int k =1; ; k++)

{

if(isPerfectNumber(k))

{

count ++;

if(count ==i) {

henrySum +=k;

}

else if(count == j)

{

henrySum +=k;

break;

}

}

}

return henrySum;

}

11. static int isDivisible(int[] a, int divisor) {

for (int i;i<a.Length;i++) {

if (a[i] % divisor != 0)

return 0;

}

return 1;

}

12. static int isNUnique(int[] a, int n) {

int countPairs = 0;

int sum = 0;

for(int i= 0; i < a.length-1; i++) {

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

sum = a[i] + a[j];

if(sum == n)

countPairs ++;

}

}

if(countPairs == 1)

return 1;

return 0;

}

12. static int isSquare(int n) {

boolean isSquare = false;

if(n < 0)

return 0;

for(int i = 1; i <= n; i++) {

if(n%i == 0) {

int multiple = n/i;

if(multiple == i)

{ isSquare = true;

break; }

}

}

if(isSquare || n == 0)

return 1;

return 0;

}

13. int isLegalNumber(int[] a, int n) {

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

if(a[i] >= n)

return 0;

}

return 1;

}

int convertToBase10(int[] a, int base) {

int sum = 0;

int multiplier = 1;

if(isLegalNumber(a, base) == 0)

return -1;

else

for(int i = a.length-1; i >= 0; i --) {

sum += a[i]\*multiplier;

multiplier \*=base;

}

return sum;

}

14. static int arrayHasNoZeroes(int[] a) {

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

if(a[i]== 0)

return 0;

}

return 1;

}

public static void main(String[] args) {

int[] arr = {1, 2, 3, 0};

System.out.println(arrayHasNoZeroes(arr));

}

14. static int computeDepth(int n) {

boolean[] flags = new boolean[10];

for (int i = 0; i < 10; i++)

flags[i] = false;

int index = 1;

while (true) {

int product = n \* index;

while (product > 0) {

int rem = product % 10;

product /= 10;

if (!flags[rem]) {

flags[rem] = true;

}

}

boolean finalFlag = true;

for (boolean aFlag : flags) {

if (!aFlag)

finalFlag = false;

}

if (finalFlag) break;

index++;

}

return index;

}

15.

static int matches(int[] a, int[] p) {

int innerLimit = 0;

int startIndex = 0;

int EndIndex = 0;

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

if(p[i] < 0)

{

innerLimit = p[i] \* -1;

EndIndex=innerLimit+startIndex;

for(int j = startIndex; j < EndIndex; j++) {

if(a[j]> 0)

return 0;

}

}

else

{

innerLimit = p[i];

EndIndex=innerLimit+startIndex;

for(int j = startIndex; j < EndIndex; j++)

{

if(a[j] < 0)

return 0;

}

}

startIndex+= innerLimit;

}

return 1;

}

16. static int isStacked(int n) {

int sum = 0;

for(int i = 1; i <= n; i++) {

sum += i;

if(sum == n) {

return 1;

}

}

return 0;

}

16.static int isSumSafe(int[] a) {

int sum = 0;

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

sum += a[i];

}

for(int j=0; j<a.length; j++) {

if(a[j] == sum)

return 0;

}

return 1;

}

16. static long isIsolated(long n) {

if((n < 0) || (n > 2097151L))

return -1;

long square = n\*n;

while(square > 0) {

long digit = square%10;

long cube = n\*n\*n;

while(cube > 0) {

long digit2 =cube%10;

if(digit == digit2)

return 0;

cube /=10;

}

square /=10;

}

return 1;

}

public static void main(String[] args) {

System.out.println(isIsolated(162));

}

18. static int isVanilla(int[] a) {

if(a.length == 0)

return 1;

int number = a[0];

while(number > 0) {

int digit = number%10;

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

int number2 = a[i];

while(number2 > 0) {

int digit2 = number2%10;

if(digit != digit2)

return 0;

number2 /= 10;

}

}

number /= 10;

}

return 1;

}

18.

//even t his does not work when {1,1,2,2}

static int isTrivalent(long[] a) {

long x = a[0];

long y = a[0];

long z = a[0];

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

if(a[i]!= x && x==y) {

y = a[i];

continue;

}

if(a[i] != x && x == z && a[i] != y) {

z = a[i];

continue;

}

if(a[i] != x && a[i] != y && a[i] != z) {

return 0;

}

}

return 1;

So Use hashset

import java.util.HashSet;

public class ThreeDifferentValues {

public static boolean containsOnlyThreeDifferentValues(int[] arr) {

// If array length is less than 3, it cannot contain 3 different values

if (arr.length < 3) {

return false;

}

// Use a set to keep track of unique values

Set<Integer> uniqueValues = new HashSet<>();

// Add elements to the set, if size exceeds 3, return false

for (int num : arr) {

uniqueValues.add(num);

if (uniqueValues.size() > 3) {

return false;

}

}

// If set size is exactly 3, return true, else return false

return uniqueValues.size() == 3;

}

public static void main(String[] args) {

int[] arr1 = {1, 2, 3, 1, 2, 3, 1, 2, 3}; // True

int[] arr2 = {1, 2, 3, 4, 5}; // False

int[] arr3 = {1, 2, 2, 2, 3, 3, 3}; // True

int[] arr4 = {1, 2}; // False

System.out.println("arr1 contains only three different values: " + containsOnlyThreeDifferentValues(arr1));

System.out.println("arr2 contains only three different values: " + containsOnlyThreeDifferentValues(arr2));

System.out.println("arr3 contains only three different values: " + containsOnlyThreeDifferentValues(arr3));

System.out.println("arr4 contains only three different values: " + containsOnlyThreeDifferentValues(arr4));

}

}

}19. static int countRepresentations(int numRupees) {

int count = 0;

for(int rupee20=0; rupee20 <= numRupees; rupee20 +=20) {

for(int rupee10=0; rupee10 <= (numRupees-rupee20); rupee10 +=10) {

for(int rupee5=0; rupee5 <=(numRupees-rupee10); rupee5 +=5) {

for(int rupee2=0; rupee2 <= (numRupees-rupee5); rupee2 +=2) {

for(int rupee1=0; rupee1 <= (numRupees-rupee2); rupee1++) {

if(rupee1+rupee2+rupee5+rupee10+rupee20 == numRupees) {

count++;

}

}

}

}

}

}

return count;

}

19. static int isSequentiallyBound(int[] a) {

for(int i=0, j=i+1; i < a.length-1; i++, j++) {

if(a[i] <=0 || a[j] <=0)

return 0;

if(a[i] > a[j])

return 0;

}

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

int count = 0;

for(int j=0; j < a.length; j++) {

if(a[i] == a[j])

count ++;

}

if(count >= a[i])

return 0;

}

return 1;

}

20. public static int isMinMaxDisjoint(int[] a) {

if (a.length <= 2) return 0;

int minValue = a[0];

int maxValue = a[0];

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

if (a[i] < minValue) {

minValue = a[i];

}

if (a[i] > maxValue) {

maxValue = a[i];

}

}

if (minValue == maxValue) return 0;

int minCount = 0;

int maxCount = 0;

for (int i = 0; i < a.length - 1; i++) {

if ((a[i] == minValue && a[i + 1] == maxValue) || (a[i] == maxValue && a[i + 1] == minValue)) {

return 0;

} else {

if (a[i] == minValue) minCount++;

if (a[i] == maxValue) maxCount++;

}

if (minCount > 1 || maxCount > 1) return 0;

}

return 1;

}

20. static int smallest(int n) {

int number = 0;

outer: for(int i=1; ;i++) {

int multiple = 1;

for(int j=1; j <= n; j++)

{ multiple = i\*j;

int digit = 0;

int count = 0;

while(multiple > 0)

{

digit = multiple%10;

if(digit == 2)

count++;

multiple /= 10;

}

if(count ==0)

continue outer;

}

number = i;

break;

}

return number;

}

21. static int[] clusterCompression(int[] a) {

int numClusters = 0;

for(int i=0, j = i+1; i < a.length-1; i++, j++) {

if(a[i] != a[j])

numClusters ++;

if(j == a.length-1)

numClusters ++;

}

int[] result = new int [numClusters];

int index = 0;

for(int i=0, j=i+1; i < a.length-1; i++, j++) {

if(a[i] != a[j]) {

result[index] = a[i];

index ++;

}

if( j == a.length-1)

result[index] = a[j];

}

return result;

}

22. static int isRailroadTie(int[] a) {

//check every zero element has two non zero neighbor

if(a[0] == 0 || a[a.length-1] == 0)

return 0;

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

if(a[i] != 0)

break;

if(i == a.length-1)

return 0;

}

for(int i=1, j=i-1, k=i+1; i < a.length-1; i++, j++, k++) {

//check every non-zero element has exactly one non- zero neighbor

if(a[i] != 0)

{

if(a[j] !=0 && a[k] != 0)

return 0;

}

//check every zero element has two non zero neighbor

if(a[i] == 0)

{

if(a[j] == 0 || a[k] == 0)

return 0;

}

}

return 1;

}

23. static int fullnessQuotient(int n) {

if(n < 0)

return -1;

int count = 0;

outer: for(int i=2; i <= 9; i++) {

int num = n;

while(num > 0) {

int digit = num % i;

if(digit == 0)

continue outer;

num /= i;

}

count ++;

}

return count;

}

24. static int isPacked(int[] a) {

int j = 0;

for (int i = 0; i < a.length; i = j) {

int count = 0;

if (a[i] <= 0)

return 0;

for (j = i; j < i + a[i]; j++) {

if (a[i] != a[j]) {

return 0;

}

}

for (int k = 0; k < a.length;k++)

{

if (a[i] == a[k])

count++;

}

if (count > a[i])

return 0;

}

return 1;

}

25. static int getExponent(int n, int p) {

if(p <= 1)

return -1;

if(n < 0)

n = -n;

int count = 0;

while(n > 0) {

if(n%p == 0)

count ++;

else

break;

n /=p;

}

return count;

}

26. static int is121Array(int[] a) {

//Check If Length of array is at least 3

if(a.length < 3)

return 0;

//Check if array starts and ends with one

if(a[0]!= 1 && a[a.length-1] != 1)

return 0;

//Check if array has only Ones and Twos and no other numbers

int count1S = 0;

int count2S = 0;

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

if(a[i] ==1) {

count1S ++;

continue;

}

if(a[i] == 2) {

count2S ++;

continue;

}

return 0;

}

if(count1S == 0 || count2S == 0)

return 0;

//Count number of Ones at beginning and at the end and compare

int count1SBefore = 0;

int count1SAfter = 0;

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

if(a[i] == 1)

{

count1SBefore++;

continue;

}

break;

}

for(int i=a.length-1; i >= 0; i--) {

if(a[i] ==1) {

count1SAfter ++;

continue;

}

break;

}

if(count1SBefore != count1SAfter)

return 0;

//Check if its only 2s Left in the middle

for(int i = count1SBefore; i < a.length-count1SAfter; i++) {

if(a[i] != 2)

return 0;

}

return 1;

}

27. static int[] filterArray(int[] a, int n) {

int num = n;

int count = 0;

int index = 0;

while(num > 0) {

int digit = num%2;

if(digit == 1)

{

count ++;

}

index ++;

num /= 2;

}

if(a.length < index)

return null;

int[] result = new int[count];

int num2 = n;

int index2 = 0;

int resultIndex = 0;

while(num2 > 0)

{

int digit = num2%2;

if(digit == 1)

{

result[resultIndex] = a[index2];

resultIndex ++;

}

index2++;

num2 /= 2;

}

return result;

}

29. static int isSequencedArray(int[] a, int m, int n) {

for(int i=0; i < a.length-1; i++) {

if(a[i] > a[i+1])

return 0;

}

for(int j=m; j <= n; j++) {

int count = 0;

for(int k=0; k < a.length; k++) {

if(j == a[k])

count++;

if(k == (a.length-1)) {

if(count = 0)

return 0;

}

}

}

return 1;

}

29. static int largestPrimeFactor(int n) {

if(n <= 1)

return 0;

int largestPrime = 0;

for(int i=2; i < n; i++) {

if(n%i == 0) {

if(isPrime(i) == 1)

largestPrime = i;

}

}

return largestPrime;

}

static int isPrime(int n) {

if(n <= 1)

return 0;

for(int i=2; i < n; i++) {

if(n%i == 0)

return 0;

}

return 1;

}

29. static int[] EncodeNumber(int a )

{

int Prime\_Factor=2;

int count=0;

int n=a;

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

while(n>1)

{

if (n%Prime\_Factor==0)

{

List.add(Prime\_Factor);

count++;

n=n/Prime\_Factor;

}

else

Prime\_Factor++;

}

int[] encodearray=new int[count];

for (int j=0;j<List.size();j++)

{

encodearray[j]=List.get(j);

}

return encodearray;

}

30. static int matchPattern(int[] a, int[] pattern) {

int i=0;

int k=0;

int matches = 0;

for(i=0; i < a.length; i++)

{

if(a[i] == pattern[k])

matches++;

else if(matches == 0 || k == pattern.length-1)

return 0;

else {

k++;

if(a[i] == pattern[k])

{

matches++;

}

else

return 0;

}

}

if(i == a.length && k == pattern.length-1)

return 1;

return 0;

}

31. static void doIntegerBasedRounding(int[] a, int n) {

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

if(n <= 0)

break;

if(a[i] < 0)

continue;

int multiple = 0;

int multiplier = 1;

int previous = 0;

while(multiple < a[i]) {

multiple = multiplier\*n;

if(multiple > a[i])

break;

previous = multiple;

multiplier ++;

}

if(previous == a[i])

a[i] = previous;

else if(a[i]-previous > multiple - a[i])

a[i] = multiple;

else if(a[i]-previous < multiple-a[i])

a[i] = previous;

else if(a[i]-previous == multiple -a[i]){

a[i] = multiple;

}

}

}

public static void main(String[] args) {

int[] arr = {5, 2,3,4,5};

doIntegerBasedRounding(arr, 3);

}

33. static int isCubePowerful(int n) {

if(n <=0)

return 0;

int sum = 0;

int num = n;

while(num > 0 ) {

int digit = num%10;

int cube = digit\*digit\*digit;

sum += cube;

num /= 10;

}

if(sum == n)

return 1;

return 0;

}

34. static int decodeArray(int[] a) {

int[] b = new int[a.length-1];

int k = 0;

for(int i=0, j=i+1; i < a.length-1; i++, j++)

{

int diff = a[i] - a[j];

if(diff < 0)

{

diff \*= -1;

}

b[k] = diff;

k++;

}

int encodedNum = 0;

int multiple10 =1;

for(int l= b.length-1; l >=0; l--) {

int digit = b[l]\* multiple10;

encodedNum += digit;

multiple10 \*= 10;

}

if(a[0] < 0)

encodedNum \*= -1;

return encodedNum;

}

34. static int isZeroPlentiful(int[] a) {

int count = 0;

int previous = 1;

int sequence = 0;

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

{

if(a[i] == 0)

{

count++;

previous = a[i];

}

if(a[i] != 0 && previous == 0) {

if(count < 4)

return 0;

sequence ++;

count = 0;

previous = a[i];

}

if(a[i] == 0 && i == a.length-1) {

if(count < 4)

return 0;

sequence ++;

}

}

return sequence;

}

35. static int isDigitIncreasing(int n) {

for (int i = 1; i <= 9; i++) {

int sum = 0;

int product = 0;

while (sum < n) {

product = (product \* 10) + i;

sum += product;

System.out.println("" + product);

}

if (sum == n)

return 1;

}

return 0;

}

36. static int decodeArray(int[] a) {

int sum = 0;

int position = 0;

for (int i = a.length - 1; i > 0; i--) {

int digit = 0;

for (int j = i; j > 0; j--) {

if (a[j - 1] != 0) {

i = j;

break;

}

digit++;

i = j;

}

for (int j = 0; j < position; j++) {

digit \*= 10;

}

sum += digit;

position++;

}

if (a[0] < 0) return -sum;

return sum;

}

36. static int isOnionArray(int[] a)

{

if (a.length == 1 || a.length ==0)

return 1;

for(int i = 0; i < a.length/2; i++) {

int j = (a.length-1) -i;

if (a[i] + a[j] > 10)

return 0;

}

return 1;

}

37. static int isPrimeHappy(int n) {

int sum = 0;

for(int i = 2; i < n; i++)

{

if(isPrime(i) == 1)

sum += i;

}

if(sum == 0)

return 0;

if(sum%n == 0)

return 1;

return 0;

}

38. [from knight]

static int[] encodeArray(int n) {

if (n == 0)

return new int[]{1};

int size = n < 0 ? 1 : 0;

int nCopy = n < 0 ? -n : n;

while (nCopy > 0) {

int digit = nCopy % 10;

nCopy /= 10;

size += digit + 1;

}

int[] result = new int[size];

nCopy = n < 0 ? -n : n;

for (int i = 0; i < size; i++) {

if (i == 0 && n < 0) {

result[i] = -1;

i++;

}

while (nCopy > 0)

{

int digit = nCopy % 10;

nCopy /= 10;

for (int j = 0; j < digit; j++)

{

result[i] = 0;

i++;

}

result[i] = 1;

i++;

}

}

return result;

}

39. static int isSystematicallyIncreasing(int[] a) {

int limit = 2;

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

for(int k=1; k < limit; k++, i++) {

// System.out.println(k);

if(a[i] != k)

return 0;

}

limit ++;

}

return 1;

}

39.

static int isFactorialPrime(int n) {

if(isPrime(n) == 1) {

for(int i=1; i < n; i++) {

if(n == factorial(i)+1)

return 1;

}

return 0;

}

return 0;

}

40. static int largestDifferenceOfEvens(int[] a) {

int largestEven = 0;

int smallestEven = 0;

int count = 0;

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

if(a[i]%2 == 0) {

count ++;

if(count ==1) {

largestEven = a[i];

smallestEven = a[i];

continue;

}

if(a[i] > largestEven)

largestEven = a[i];

if(a[i] < smallestEven)

smallestEven = a[i];

}

}

if(count <2)

return -1;

return (largestEven - smallestEven);

}

40. static int isHodder(int n) {

if(isPrime(n) == 1) {

n +=1;

while(n > 1) {

if(n%2 != 0)

return 0;

n /= 2; n=n/2;

}

return 1;

}

return 0;

}

40. static int areAnagrams(char[] a1, char[] a2) {

if (a1.length != a2.length)

return 0;

char[] a1Copy = new char[a1.length];

char[] a2Copy = new char[a2.length];

for (int i = 0; i < a1.length; i++) {

a1Copy[i] = a1[i];

a2Copy[i] = a2[i];

}

for (int i = 0; i < a1Copy.length; i++) {

for (int j = 0; j < a2Copy.length; j++) {

if (a1Copy[i] == a2Copy[j]) {

a1Copy[i] = ' ';

a2Copy[j] = ' ';

}

}

}

for (int i = 0; i < a1Copy.length; i++) {

if (a1Copy[i] != ' ' || a2Copy[i] != ' ')

return 0;

}

return 1;

}

}

42.static int isTriangular(int n) {

int sum = 0;

int num = 1;

while(sum < n) {

sum += num;

num ++;

}

if(sum == n)

return 1;

return 0;

}

42. static int isMercurial(int [] a) {

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

if(a[i] == 1) {

for(int j=i+1; j < a.length; j++, i++) {

if(a[j] == 3) {

for(int k=j+1; k < a.length; k++, j++, i++) {

if(a[k] == 1)

return 0;

}

}

}

}

}

return 1;

}

43. static int is235Array(int[] a) {

int countDivisible2 = 0;

int countDivisible3 = 0;

int countDivisible5 = 0;

int countNotDivisible = 0;

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

if(a[i]%2 == 0)

countDivisible2 ++;

if(a[i]%3 == 0)

countDivisible3 ++;

if(a[i]%5 == 0)

countDivisible5 ++;

if(a[i]%2 !=0 && a[i]%3 != 0 && a[i]%5 != 0)

countNotDivisible ++;

}

if((countDivisible2+countDivisible3+countDivisible5+countNotDivisible) == a.length)

return 1;

return 0;

}

48. static int isNPrimeable(int[] a, int n) {

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

if(isPrime(a[i]+n) == 0)

return 0;

}

return 1;

}

48. static int is121Array(int[] a) {

//Check If Length of array is at least 3

if(a.length < 3)

return 0;

//Check if array starts and ends with one

if(a[0]!= 1 && a[a.length-1] != 1)

return 0;

//Check if array has only Ones and Twos and no other numbers

int count1S = 0;

int count2S = 0;

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

if(a[i] ==1) {

count1S ++;

continue;

}

if(a[i] == 2) {

count2S ++;

continue;

}

return 0;

}

if(count1S == 0 || count2S == 0)

return 0;

//Count number of Ones at beginning and at the end and compare

int count1SBefore = 0;

int count1SAfter = 0;

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

if(a[i] == 1) {

count1SBefore++;

continue;

}

break;

}

for(int i=a.length-1; i >= 0; i--) {

if(a[i] ==1) {

count1SAfter ++;

continue;

}

break;

}

if(count1SBefore != count1SAfter)

return 0;

//Check if only 2s Left in the middle

for(int i = count1SBefore; i < a.length-count1SAfter; i++) {

if(a[i] != 2)

return 0;

}

return 1;

}

49. int isSquare(int n) {

boolean isSquare = false;

if(n < 0)

return 0;

for(int i = 1; i <= n; i++) {

if(n%i == 0) {

int multiple = n/i;

if(multiple == i)

isSquare = true;

if(isSquare)

break;

}

}

if(isSquare || n == 0)

return 1;

return 0;

}

50. static int isComplete(int[] a) {

if(a.length < 2)

return 0;

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

if(a[i]%2 == 0)

break;

if(i == a.length-1)

return 0;

}

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

if(isPerfectSquare(a[i]) == 1)

break;

if(i == a.length-1)

return 0;

}

for(int i=0; i < a.length-1; i++) {

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

if(a[i]+a[j] == 8)

return 1;

}

}

return 0;

}

static int isPerfectSquare(int n) {

for(int i=1; i <= n; i++) {

if(n%i == 0) {

int num = n/i;

if(num == i)

return 1;

}

}

return 0;

}

50. static int loopSum(int[] a, int n) {

int i = 0;

int sum = 0;

int count = 0;

for(i =0; i < a.length; )

{

sum += a[i];

count ++;

if(count == n)

break;

if(i == a.length-1 && count != n) {

i = 0;

continue;

}

i++;

}

return sum;

}

51. static int allValuesTheSame(int[] a) {

for(int i=0, j=i+1; j < a.length; j++) {

if(a[i] != a[j])

return 0;

}

return 1;

}

51. static int hasNValues(int[] a, int n) {

int[] arr = new int[n];

int count = 0;

int index = 0;

outer: for(int i= 0; i < a.length; i++)

{

for(int j=0; j <count; j++)

{

if(a[i] == arr[j]) //this is getting same value in the arr array so start over ourter loop

continue outer;// it shall start outer loop for next i

}

count ++;

if(count > n)

return 0;

arr[index] = a[i];

index ++;

}

if(count < n)

return 0;

return 1;

}

52 static int sameNumberOfFactors(int n1, int n2) {

if(n1 < 0 || n2 < 0)

return -1;

int count1 = 0;

int count2 = 0;

for(int i=1; i <= n1; i++) {

if(n1%i == 0)

count1 ++;

}

for(int i=1; i <= n2; i++) {

if(n2%i == 0)

count2 ++;

}

if(count1 == count2)

return 1;

return 0;

}

52. static double eval(double x, int[] a) {

double sum = 0d;

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

double num = 1.0d;

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

{

num \*= x;

}

sum += a[i]\*num;

}

return sum;

}

53. public static int isLayered(int[]a){

int count=0;

for(int i=0;i<a.length-1;i++){

if(a[i]>a[i+1]){

return 0;

}

}

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

count=0;

for(int j=0;j<a.length;j++){

if(a[i]==a[j]){

count++;

}

}

if(count<2){

return 0;

}

}

return 1;

}

55. static void updateMileageCounter(int[] a, int miles) {

for (int i = 0; i < a.length && miles > 0; i++) {

miles = a[i] + miles;

if (miles > 9) {

int tempMiles = miles % 10;

miles /= 10;

a[i] = tempMiles;

} else {

a[i] = miles;

miles = 0;

}

}

}

}57. static int isConsecutiveFactored(int n) {

int previous = 1;

for(int i=2; i < n; i++) {

if(n%i == 0)

{

if(i == (previous+1))

return 1;

previous = i;

}

}

return 0;

}

57. static int isTwinPrime(int n) {

if(isPrime(n) == 1) {

if(isPrime(n+2)==1 || isPrime(n-2)==1)

return 1;

}

return 0;

}

static int isPrime(int n) {

if(n < 2)

return 0;

for(int i=2; i < n; i++) {

if(n%i ==0)

return 0;

}

return 1;

}

59. static int largestAdjacentSum(int[] a) {

int sum = 0;

int largestSum = 0;

for(int i = 0; i <a.length-1; i++) {

sum = a[i] + a[i+1];

if(sum> largestSum)

largestSum = sum;

}

return largestSum;

}

59. static int isZeroBalanced(int[] a) {

if(a.length == 0)

return 0;

int sum = 0;

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

sum += a[i];

}

if(sum != 0)

return 0;

outer:for(int i=0; i < a.length/2; i++) {

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

if(a[i] > 0)

{

int num = a[i] \* -1;

if(num == a[j])

continue outer;

if(j == a.length-1)

return 0;

}

if(a[i] < 0)

{

int num = a[i] \* -1;

if(num == a[j])

continue outer;

if(j == a.length-1)

return 0;

}

}

}

return 1;

}

60. static int findSmallestBEQnumber() {

int num = 0;

for(int i=1; ;i++) {

int count = 0;

long cube = i \* i \* i;

while(cube > 0) {

long digit = cube%10;

if(digit == 6)

count ++;

cube /= 10;

}

if(count == 4) {

num = i;

break;

}

}

return num;

}

60. static int decider(int i,int Len)

{

for (int x = 1; x < Len; x += 3)

{

if (i==x)

return 0;

}

return 1;

}

static int isZeroLimited(int[] a)

{

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

{

if ( decider(i,a.length )==0)

{

if (a[i]!=0)

return 0;

}

else if( decider(i,a.length )==1)

{ if (a[i]==0)

return 0;

}

}

return 1;

}

62. static int isCubePerfect(int[] a) {

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

if(isCube(a[i]) == 0)

return 0;

}

return 1;

}

static int isCube(int n) {

int num = n;

if( num < 0)

num \*= -1;

for(int i = 0; i <= num; i++) {

int cube = i \* i \* i;

if(cube == num)

return 1;

if(cube > num)

break;

}

return 0;

}

62. static int countOnes(int n) {

int count = 0;

int num = n;

while(num > 0) {

int digit = num%2;

if(digit == 1)

count ++;

num /= 2;

}

return count;

}

62. static int isDaphne(int[] a) {

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

if(a[0]%2 == 0) {

if(a[i]%2 != 0)

return 0;

}

if(a[0]%2 != 0) {

if(a[i]%2 == 0)

return 0;

}

}

return 1;

}

63. static int isOddValent(int[] a) {

outer:for(int i=0; i < a.length-1; i++) {

int count = 0;

for(int j =i; j< a.length; j++) {

if(a[i] == a[j])

count ++;

if(count > 1)

break outer;

}

if(i == a.length-2)

return 0;

}

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

if(a[i]%2 != 0)

return 1;

}

return 0;

}

63. static int isNormal(int n) {

for(int i =2; i < n; i++) {

if(n%i == 0) {

if(i%2 == 1)

return 0;

}

}

return 1;

}

64. static int isAllPossibilities(int[] a) {

if(a.length == 0)

return 0;

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

for(int j=0; j < a.length; j++) {

if(a[j] == i)

break;

if(j == a.length-1)

return 0;

}

}

return 1;

}

64. static int isFilter(int[] a) {

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

if(a[i] == 9) {

for(int j=0; j < a.length; j++) {

if(a[j] == 11)

break;

if(j == a.length-1)

return 0;

}

}

if(a[i] ==7) {

for(int j=0; j < a.length; j ++) {

if(a[j] == 13)

return 0;

}

}

}

return 1;

}

65.

static int isFineArray(int[] a) {

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

if(isPrime(a[i]) == 1) {

int x = a[i]+2;

int y = a[i]-2;

if(isPrime(x)==1 || isPrime(y)==1) {

for(int j=0; j < a.length; j++) {

if(a[j] == x || a[j] == y)

break;

if(j == a.length-1)

return 0;

}

}

}

}

return 1;

}

65. static int isDigitSum(int n, int m) {

if(n < 0)

return -1;

int num = n;

int sum = 0;

while(num > 0) {

int digit = num % 10;

sum += digit;

num /= 10;

}

if(sum < m)

return 1;

return 0;

}

66. static int isEvens(int n) {

int num = n;

while(num > 0) {

int digit = num%2;

if(digit%2 != 0)

return 0;

num /= 10;

}

return 1;

}

66. static int isMagicArray(int[] a) {

int sum = 0;

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

if(isPrime(a[i])==1)

sum += a[i];

}

if(a[0] == sum)

return 1;

return 0;

}

67. static int isComplete(int[] a) {

int min = a[0];

int max = a[0];

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

if(a[i]%2 ==0) {

min = a[i];

max = a[i];

break;

}

if(i == a.length-1)

return 0;

}

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

if(a[i]%2 == 0) {

if(a[i] > max)

max = a[i];

if(a[i] < min)

min = a[i];

}

}

if(max == min)

return 0;

for(int i= min+1; i < max; i++) {

for(int j=0; j < a.length; j++) {

if(i == a[j])

break;

if(j == a.length-1)

return 0;

}

}

return 1;

}

67. static int isPrimeProduct(int n) {

if (n <= 1)

return 0;

boolean primeProduct = false;

for (int factor = 2; factor < n; factor++) {

if (n % factor == 0) {

int nextFactor = n / factor;

if (nextFactor != factor && isPrime(factor) && isPrime(nextFactor)) {

return 1 ; }

}

}

return 0;

}

68. static int isBalanced(int[] a) {

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

if(i%2 == 0) {

if(a[i]%2 == 1)

return 0;

}

else if(i%2 == 1) {

if(a[i]%2 == 0)

return 0;

}

}

return 1;

}

68. static int isCentered(int[] a) {

if(a.length%2 == 0)

return 0;

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

if(a[i] <= a[a.length/2] && i != a.length/2)

return 0;

}

return 1;

}

69. static boolean hasKSmallFactors(int k, int n) {

if (k <0 || n <0)

return false;

for(int i = 1; i < k; i ++) {

if(n%i == 0)

{

int result = n/i;

if(result < k && i<k)

//break;

return true ;

if(i == k-1)

return false;

}

}

return false;

}

69. static int[] fill(int[] arr, int k, int n) {

if(k < 1 || n < 1)

return null;

int[] arr2 = new int[n];

for(int i=0; i < arr2.length; )

{

for(int j=0; j < k; j++)

{

arr2[i] = arr[j];

i ++;

if(i > arr2.length-1)

break;

}

}

return arr2;

}

70. private static int isHollow(int[] array) {

int length = array.length;

int startCount = 0;

int endCount = 0;

int zeroCount = 0;

int nonZeroCount = 0;

if (array[0] == 0 || array[length - 1] == 0) return 0;

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

if (array[i] != 0) {

startCount++;

} else {

break;

}

}

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

if (array[i] == 0) {

zeroCount++;

} else {

nonZeroCount++;

}

}

for (int i = length - 1; i >= 0; i--) {

if (array[i] != 0) {

endCount++;

} else {

break;

}

}

if (startCount == endCount && (startCount + endCount) == nonZeroCount && zeroCount >= 3) {

return 1;

}

return 0;

}

70. static int minDistance(int n) {

int min = n - 1;

int diff = 0;

int previous = 1;

for(int i=2; i <= n; i++) {

if(n%i == 0) {

diff = i - previous;

if(diff < min)

min = diff;

previous = i;

}

}

return min;

}

70. static int isWave(int[] a) {

for(int i=0; i < a.length-1; i++) {

if(a[i]%2 == 0) {

if(a[i+1]%2 == 0)

return 0;

}

else if(a[i]%2 == 1) {

if(a[i+1]%2 == 1)

return 0;

}

}

return 1;

}

71. static int isBean(int[] a) {

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

if(a[i] == 9) {

for(int j=0; j < a.length; j++) {

if(a[j] == 13)

break;

if(j == a.length-1)

return 0;

}

}

if(a[i] == 7) {

for(int j=0; j < a.length; j++) {

if(a[j] == 16)

return 0;

}

}

}

return 1;

}

71. static int countDigit(int n, int digit) {

if(n < 0 || digit < 0)

return -1;

int num = n;

int count = 0;

while(num > 0) {

if(num%10 == digit)

count++;

num /= 10;

}

return count;

}

72. static int isBunkerArray(int[] a) {

for(int i=0; i < a.length-1; i++) {

if(a[i]%2 == 1) {

if(isPrime(a[i+1])==1)

return 1;

}

}

return 0;

}

72. static int isMeera(int[] a) {

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

int num = a[i]\*2;

for(int j=0; j < a.length; j++) {

if(a[j] ==num)

return 0;

}

}

return 1;

}

73. static int isMeera(int n) {

int count = 0;

for(int i=2; i < n; i++ ) {

if(n%i == 0)

count ++;

}

if (n%count==0)

return 1;

return 0;

}

73.

static int isBunker(int[] a) {

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

if(isPrime(a[i])==1) {

for(int j=0; j < a.length; j++) {

if(a[j] ==1)

return 1;

}

}

if(a[i] == 1) {

for(int j=0; j < a.length; j++) {

if(isPrime(a[j]) ==1)

return 1;

}

}

}

return 0;

}

74. static int isContinuousFactored(int n) {

int prev = 1;

for(int i=2; i < n; i++)

{

if(n%i == 0)

{

prev = i;

int mult = 1;

mult \*= i;

for(int j= i+1; j < n; j++)

{

if(n%j == 0)

{

if((j-prev) == 1)

{

mult \*= j;

if(mult == n)

return 1;

if(mult > n)

break;

prev = j;

}

}

else

break;

}

}

}

return 0;

}

74. static int isContinuousFactored(int n) {

for (int i = 2; i < n; i++)

{

if (n % i == 0 && n % (i + 1) == 0)

return 1;

}

return 0;

}

75. public class IsSetEqual {

static int isSetEqual(int[] a, int[] b) {

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

for(int j=0; j < b.length; j++) {

if(a[i] == b[j])

break;

if(j == b.length-1)

return 0;

}

}

return 1;

}

75. static int isTwin(int[] a) {

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

if(isPrime(a[i]) == 1) {

int n1 = a[i] + 2;

int n2 = a[i] - 2;

if(isPrime(n1) == 1 || isPrime(n2) == 1) // if twin prime

{

for(int j= 0; j < a.length; j++) {

if((a[j] == n1 && isPrime(n1)==1) || (a[j] == n2 && isPrime(n2)==1))

break;

if(j == a.length-1)

return 0;

}

}

}

}

return 1;

}

75. static int isSmart(int n) {

int num = 1;

int count = 1;

while(num <=n) {

if(num == n)

return 1;

num += count;

count++;

}

return 0;

}

76. static int isNiceArray(int[] a) {

int sum = 0;

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

if(isPrime(a[i])==1)

sum += a[i];

}

if(sum == a[0])

return 1;

return 0;

}

76. static int isComplete(int[] a) {

int maxEven = 0;

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

if(a[i] < 0)

return 0;

if(a[i]%2 == 0)

{ if(a[i] > maxEven)

maxEven = a[i];

}

}

for(int i=2; i < maxEven; i +=2) {

for(int j=0; j < a.length; j++) {

if(a[j] == i)

break;

if(j == a.length-1)

return 0;

}

}

return 1;

}

77. static int factorEqual(int n, int m) {

int count1 = 0;

int count2 = 0;

for(int i=1; i <=n ; i++) {

if(n%i == 0)

count1 ++;

}

for(int i=1; i <=m ; i++) {

if(m%i == 0)

count2 ++;

}

if(count1 == count2)

return 1;

return 0;

}

77. static int isMeera(int[] a) {

int sum = 0;

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

if(a[i] >= i)

return 0;

sum += a[i];

}

if(sum == 0)

return 1;

return 0;

}

78. static int isTriple(int[] a) {

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

int count = 0;

for(int j=0; j < a.length; j++) {

if(a[i] == a[j])

count ++;

if(j == a.length-1) {

if(count != 3)

return 0;

}

}

}

return 1;

}

78. static int isFibonacci(int n) {

int previous = 1;

int next = 1;

while(next < n) {

int temp = next;

next = previous + next;

previous = temp;

}

if(next == n)

return 1;

return 0;

}

78. static int isMeera(int[] a) {

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

if(a[i] == 0) {

for(int j=0; j < a.length; j++) {

if(isPrime(a[j]) == 1)

return 1;

if(j == a.length-1)

return 0;

}

}

if(isPrime(a[i]) == 1) {

for(int j=0; j < a.length; j++) {

if(a[j] == 0)

return 1;

if(j == a.length-1)

return 0;

}

}

}

return 1;

}

static int isPrime(int n) {

if(n < 2)

return 0;

for(int i=2; i < n; i++) {

if(n%i == 0)

return 0;

}

return 1;

}

79. static int isFancy(int n) {

int before = 1;

int previous = 1;

int fancy = 1;

while(fancy < n) {

fancy = 3\*previous + 2\*before;

if(fancy == n)

return 1;

before = previous;

previous = fancy;

}

return 0;

}

80. static int isMeera(int[] a) {

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

if(a[i] == 1) {

for(int j=0; j < a.length; j++) {

if(a[j] == 9)

return 1;

if(j == a.length-1)

return 0;

}

}

if(a[i] == 9) {

for(int j=0; j < a.length; j++) {

if(a[j] == 1)

return 1;

if(j == a.length-1)

return 0;

}

}

}

return 1;

}

81. static int isDual(int[] a) {

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

int count = 0;

for(int j=0; j < a.length; j++) {

if(a[i] == a[j])

count ++;

if(j == a.length-1) {

if(count != 2)

return 0;

}

}

}

return 1;

}

82. static int isBalanced(int[] a) {

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

if(i%2 == 0) {

if(a[i]%2 == 1)

return 0;

}

else if(i%2 == 1) {

if(a[i]%2 == 0)

return 0;

}

}

return 1;

}

82. static int isOddHeavy(int[] a) {

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

if(a[i]%2 !=0)

break;

if(i == a.length-1)

return 0;

}

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

if(a[i]%2 != 0) {

for( int j= 0; j < a.length; j++) {

if(a[j]%2 == 0) {

if(a[j] > a[i])

return 0;

}

}

}

}

return 1;

}

82. static int isNormal(int n) {

for(int i =2; i < n; i++) {

if(n%i == 0) {

if(i%2 == 1)

return 0;

}

}

return 1;

}

83. static int isAllPossibilities(int[] a) {

if(a.length == 0)

return 0;

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

for(int j=0; j < a.length; j++) {

if(a[i] == j)

break;

if(j == a.length-1)

return 0;

}

}

return 1;

}

83. static int isDual(int[] a) {

if(a.length%2 == 1)

return 0;

for(int i=0, j=2; j < a.length-1; i +=2, j +=2) {

int sum1 = a[i] + a[i+1];

int sum2 = a[j] + a[j+1];

if(sum1 != sum2)

return 0;

}

return 1;

}

85. static int goodSpread(int[] a) {

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

int count = 0;

for(int j=0; j < a.length; j++) {

if(a[i] == a[j])

count ++;

if(j == a.length-1) {

if(count > 3)

return 0;

}

}

}

return 1;

}

86. static int sumDigits(int n) {

int sum = 0;

int num = n;

while(num > 0) {

int digit = num%10;

sum += digit;

num /= 10;

}

return sum;

}

86.

static int isMeera(int[] a) {

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

{

if (a[i] >= i)

return 0;

}

return 1;

}

87. static int isBeanArray(int[] a) {

if (a == null || a.length==0) return 0;

int a1 = a[0];

int sum = 0;

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

{

if (isPrime(a[i])) {

sum += a[i];

}

}

if (sum != a1) return 0;

else return 1;

}

87. static int isComplete(int[] a) {

int maximumEven=0;

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

if (a[i] < 0) {

return 0;

}

if (a[i] % 2 == 0 && a[i] > maximumEven) {

maximumEven = a[i];

}

}

for (int i = 2; i < maximumEven; i++) {

if (i % 2 == 0) {

boolean flag = false;

for (int j=0; j< a.length ;j++) {

if (a[j] == i)

{

flag = true;

break;

}

}

if (!flag)

return 0;

}

}

return 1;

85. static int isDaphne(int[] a) {

int countBegin = 0;

int countEnd = 0;

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

if(a[i]%2 == 1)

break;

if(i == a.length-1)

return 0;

}

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

if(a[i]%2 == 0)

countBegin ++;

if(a[i]%2 == 1)

break;

}

for(int i=a.length-1; i >= 0; i--) {

if(a[i]%2 == 0)

countEnd ++;

if(a[i]%2 == 1)

break;

}

if(countBegin == countEnd)

return 1;

return 0;

}

85.

static int factorTwoCount(int n) {

int count = 0;

int num = n;

while(num%2 ==0) {

count ++;

num /= 2;

}

return count;

}

89.

public class IsFilter {

static int isFilter(int[] a) {

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

if(a[i] == 9) {

for(int j=0; j < a.length; j++) {

if(a[j] == 11)

break;

if(j == a.length-1)

return 0;

}

}

if(a[i] ==7) {

for(int j=0; j < a.length; j ++) {

if(a[j] == 13)

return 0;

}

}

}

return 1;

}

public static void main(String[] args) {

System.out.println(isFilter(new int[] {4, 7, 13}));

}

}

89.

public class IsDual {

static int isDual(int[] a) {

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

int count = 0;

for(int j=0; j < a.length; j++) {

if(a[i] == a[j])

count ++;

if(j == a.length-1) {

if(count != 2)

return 0;

}

}

}

return 1;

}

public static void main(String[] args) {

System.out.println(isDual(new int[] {3, 1, 1, 2, 2}));

}

}

89.

static int isBunker(int n) {

int previousElement = 1;

int index = 0;

int element = 0;

while (element < n) {

element = previousElement + index;

previousElement = element;

index++;

}

if (element == n) return 1;

return 0;

}

public static void main(String[] args) {

System.out.println(isBunker(11));

}

90.

package practice;

public class IsMeera4 {

static int isMeera(int[] a) {

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

if(a[i] == 0)

{

for(int j=0; j < a.length; j++)

{

if(isPrime(a[j]) == 1)

return 1;

if(j == a.length-1)

return 0;

}

}

if(isPrime(a[i]) == 1)

{

for(int j=0; j < a.length; j++) {

if(a[j] == 0)

return 1;

if(j == a.length-1)

return 0;

}

}

}

return 1;

}

static int isPrime(int n) {

if(n < 2)

return 0;

for(int i=2; i < n; i++) {

if(n%i == 0)

return 0;

}

return 1;

}

public static void main(String[] args) {

System.out.println(isMeera(new int[] {3, 7, 0, 8, 0, 5} ));

}

}

90.

package practice;

public class IsFibonacci {

static int isFibonacci(int n) {

int previous = 1;

int next = 1;

while(next < n) {

int temp = next;

next = previous + next;

previous = temp;

}

if(next == n)

return 1;

return 0;

}

public static void main(String[] args) {

System.out.println(isFibonacci(13));

}

}

public class IsBean2 {

static int isBean(int[] a) {

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

int n = a[i] + 1;

int m = a[i] - 1;

for(int j=0; j < a.length; j++) {

if(a[j] == n || a[j] ==m)

break;

if(j == a.length-1)

return 0;

}

}

return 1;

}

public static void main(String[] args) {

System.out.println(isBean(new int[] {3, 4, 5, 7}));

}

}

Extras1:

public class RapidIncrease {

public static void main(String[] args) {

System.out.println(isRapidlyIncreasing(new int[]{1,3,9,27}));

System.out.println(isRapidlyIncreasing(new int[]{1,3,200,500}));

System.out.println(isRapidlyIncreasing(new int[]{1}));

System.out.println(isRapidlyIncreasing(new int[]{1,3,9,26}));

System.out.println(isRapidlyIncreasing(new int[]{1,3,7,26}));

System.out.println(isRapidlyIncreasing(new int[]{1,3,8,26}));

}

public static int isRapidlyIncreasing(int[] nums) {

if (nums.length == 1) return 1;

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

int sum = 0, j = i - 1;

while(j >= 0) {

sum += nums[j];

j--;

}

if (nums[i] <= 2 \* sum) {

return 0;

}

}

return 1;

}

}

static int[] solve10() {

int factorialX = 1;

int factorialY = 1;

int x = 1;

int y = 1;

boolean matched = false;

int tenFactorial = 1;

int[] result = new int[2];

for (int i = 1; i <= 10; i++) {

tenFactorial \*= i;

}

for (x = 1; x <= 10 && !matched; x++) {

factorialX \*= x;

factorialY = 1;

for (y = 1; y <= 10 && !matched; y++) {

factorialY \*= y;

if (tenFactorial == factorialX + factorialY) {

matched = true;

break;

}

}

}

if (matched) {

result[0] = x;

result[1] = y;

} else {

result[0] = 0;

result[1] = 0;

}

return result;

}

}

public class test {

public static void main(String[] args) {

List<Employee> empList=new ArrayList<>();

empList.add(new Employee("ID1","Anol","HR",40));

empList.add(new Employee("ID2","Anol","UY",40));

empList.add(new Employee("ID3","Neil","Accounts",10));

Map<String,Long> EmpCount= empList.stream().collect(Collectors.groupingBy(

Employee :: getName,Collectors.counting()));

System.out.println(EmpCount);

Map<String, Integer> map=new HashMap<>();

for( Employee e: empList)

{

if (!map.containsKey(e.getName()))

{

map.put(e.getName(),1);

}

else

map.put(e.getName(),map.get(e.getName())+1);

}

System.out.println(map);

}

}

import java.io.\*;

import java.util.Scanner;

class GFG {

public static void main (String[] args) {

Scanner scanner = new Scanner(System.in);

String Str = scanner.nextLine();

char[] arr = Str.toCharArray();

String rev = "";

for(int i = Str.length() - 1; i >= 0; i--)

{

rev = rev + Str.charAt(i);

}

System.out.println(rev);

}

}

// java program to reverse a word

import java.io.\*;

import java.util.Scanner;

class GFG {

public static void main (String[] args) {

String str= "Geeks", nstr="";

char ch;

System.out.print("Original word: ");

System.out.println("Geeks"); //Example word

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

{

ch= str.charAt(i); //extracts each character

nstr= ch+nstr; //adds each character in front of the existing string

}

System.out.println("Reversed word: "+ nstr);

}

}

public class ReverseNumber {

static int reverseNumber(int n) {

int sign = 1;

if(n == 0)

return 0;

if(n < 0) {

sign = -1;

n = -n;

}

int reverse = 0;

while(n != 0) {

reverse = (reverse\*10) + (n%10);

n /= 10;

}

return reverse\*sign;

}