public NewClass() {

static int method(String s) {

String[] ar = {"sumDiffrence", "fc", "hasNValues", "equivalentArrays", "isNPrimeable", "is121Array", "isDigitIncreasing", "pairwiseSum", "isMartian", "isOlympic", "isTwinPaired", "isRapidlyIncreasing", "is123Array", "sameNumberOfFactors", "computeWeightedSum", "is235Array", "closestFibonacci", "isOnionArray", "isMercurial", "isCenteredArray", "isOddEvenDifference", "reverse", "largestAdjacentSum", "checkConcatenatedSum", "isSequencedArray"};

String[] ar2 = {"largestPrimeFactor", "encodeNumber", "matchPattern", "doIntegerBasedRounding", "isCubePowerful", "isZeroPlentiful", "isPrimeHappy", "encodeArray", "isFactorialPrime", "isHodder", "areAnagrams","isPairedN"};

isBrownian(int n), int isOneHeavy(int[] a)

for (String i : ar) {

if (s == i) {

return 1;

}

}

for (String i : ar2) {

if (s == i) {

return 1;

}

}

return 0;

}

static int sumDiffrence(int[] a) {

int x = 0;

int y = 0;

int l = a.length;

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

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

x = x + a[i];

} else {

y = y + a[i];

}

}

return x - y;

}

static char[] fc(char[] a, int start, int len) {

if (start + len > a.length || start < 0 || len < 0) {

return null;

}

char[] ar = new char[len];

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

ar[i] = a[start + i];

}

return ar;

}

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

if (a.length < n) {

return 0;

}

int num = 0;

int[] array = new int[a.length];

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

int count = 0;

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

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

count = count + 1;

}

}

if (count == 0) {

array[num] = a[i];

num++;

}

}

if (num == n) {

return 1;

} else {

return 0;

}

}

static int equivalentArrays(int[] a1, int[] a2) {

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

int count = 0;

for (int j = 0; j < a2.length; j++) {

if (a1[i] == a2[j]) {

count = 1;

}

}

if (count == 0) {

return 0;

}

}

for (int i = 0; i < a2.length; i++) {

int count = 0;

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

if (a2[i] == a1[j]) {

count = 1;

}

}

if (count == 0) {

return 0;

}

}

return 1;

}

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

int al = a.length;

int c = 0;

if (al == 0) {

return 1;

} else {

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

int num = a[i] + n;

int jnum = num - 1;

if (num == 1 || num == 0) {

continue;

} else {

for (int j = 2; j < jnum; j++) {

if (num % j == 0) {

return 0;

}

}

}

}

return 1;

}

}

static int is121Array(int[] a) {

int len = a.length;

int count = 0;

int idx = 1;

int k = 0;

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

return 0;

} else {

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

if (a[i] != a[len - 1 - i]) {

return 0;

}

if (a[i] != 1 && a[i] != 2) {

return 0;

} else if (a[i] == 1) {

if (idx == 1) {

count++;

} else {

k = 1;

}

} else if (a[i] == 2) {

idx = 2;

if (k == 1) {

return 0;

}

}

}

if (idx == 1) {

return 0;

}

return 1;

}

}

static int[] isDigitIncreasing(int[] a) {

int al = a.length;

if (al <= 2 || al % 2 == 1) {

return null;

} else {

int c = 0;

int n = al / 2;

int[] ar = new int[n];

for (int i = 0; i < al; i += 2) {

ar[c] = a[i] + a[i + 1];

c++;

}

return ar;

}

}

static int[] pairwiseSum(int[] a) {

int al = a.length;

if (al <= 2 || al % 2 == 1) {

return null;

} else {

int c = 0;

int n = al / 2;

int[] ar = new int[n];

//int[ ] ar=new int[n];

for (int i = 0; i < al; i = i + 2) {

ar[c] = a[i] + a[i + 1];

}

c++;

return ar;

}

}

static int isMartian(int[] a) {

int l = a.length;

if (l == 0) {

return 0;

} else if (l == 1 && a[0] != 1) {

return 0;

} else if (l == 1 && a[0] == 1) {

return 1;

} else {

int max = 0;

int min = 0;

for (int i = 0; i < l - 1; i++) {

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

return 0;

} else if (a[i] == 1) {

max += 1;

} else if (a[i] == 2) {

min += 1;

}

}

if (a[l - 1] == 1) {

max += 1;

} else if (a[l - 1] == 2) {

min += 1;

}

if (max > min) {

return 1;

} else {

return 0;

}

}

}

static int isOlympic(int[] a) {

int len = a.length;

if (len == 0) {

return 0;

}

if (len == 1 && a[0] >= 0) {

return 1;

}

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

int sum = 0;

if (a[i] < 0) {

return 0;

}

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

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

sum += a[j];

}

}

if (a[i] < sum) {

return 0;

}

}

return 1;

}

static int isTwinPaired(int[] a) {

int len = a.length;

if (len == 0) {

return 1;

} else {

int ce = 0;

int ca = 0;

int even = 0;

int odd = 0;

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

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

if (ce == 1 && odd > a[i]) {

return 0;

} else {

ce = 1;

}

odd = a[i];

} else {

if (ca == 1 && even > a[i]) {

return 0;

} else {

ca = 1;

}

even = a[i];

}

}

return 1;

}

}

static int isRapidlyIncreasing(int[] a) {

int len = a.length;

int sum = a[0];

if (len == 1) {

return 1;

} else {

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

if (2 \* sum >= a[i]) {

return 0;

}

sum = sum + a[i];

}

return 1;

}

}

static int is123Array(int[] a) {

int len = a.length;

if (len == 0 || len % 3 != 0) {

return 0;

} else {

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

if ((i + 1 )% 3 == 1 && a[i] != 1) {

return 0;

} else if ((i + 1 )% 3 == 2 && a[i] != 2) {

return 0;

} else if ((i + 1) % 3 == 0&& a[i] != 3) {

return 0;

}

}

return 1;

}

}

static int sameNumberOfFactors(int n1, int n2) {

if (n1 < 0 || n2 < 0) {

return -1;

} else {

int cn1 = 0;

int cn2 = 0;

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

if (n1 % i == 0) {

cn1++;

}

}

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

if (n2 % i == 0) {

cn2++;

}

}

if (cn1 == cn2) {

return 1;

} else {

return 0;

}

}

}

static int computeWeightedSum(int[] a) {

int len = a.length;

int sumEven = 0;

int sumOdd = 0;

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

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

sumEven += a[i];

} else {

sumOdd += a[i];

}

}

return 2 \* sumEven + 3 \* sumOdd;

}

static int is235Array(int[] a) {

int len = a.length;

if (len == 0) {

return 1;

} else {

int c2 = 0;

int c3 = 0;

int c5 = 0;

int cn = 0;

int cy = 0;

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

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

c2++;

}

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

c3++;

}

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

c5++;

}

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

cn++;

}

}

int sum = c2 + c3 + c5 + cn;

if (len == sum) {

return 1;

} else {

return 0;

}

}

}

static int closestFibonacci(int n) {

if (n == 1 || n == 2 || n == 3) {

return n;

} else if (n < 1) {

return 0;

} else {

int m1 = 1;

int m2 = 2;

int fab = 0;

while (fab <= n) {

fab = m1 + m2;

m1 = m2;

m2 = fab;

};

return m1;

}

}

static int isOnionArray(int[] a) {

int len = a.length;

if (len < 2) {

return 1;

} else {

int m = len / 2; // m is middle indexx whre the condition j==k

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

int k = len - 1 - j;

if (a[j] + a[k] > 10) {

return 0;

}

}

return 1;

}

}

static int isMercurial(int[] a) {

if (a.length == 0) {

return 1;

}

int c1 = 0, c3 = 0;

for (int i : a) {

if (i == 1 && c1 == 0) {

c1 = 1;

} else if (i == 3 && c1 == 1) {

c3 = 1;

} else if (i == 1 && c3 == 1) {

return 0;

}

}

return 1;

}

static int isCenteredArray(int[] a) {

int len = a.length;

if (len % 2 == 0) {

return 0;

} else if (len == 1) {

return 1;

} else {

int m = len / 2;

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

if (a[1] <= a[m] || a[len - 1 - i] <= a[m]) {

return 0;

}

}

}

return 1;

}

static int isOddEvenDifference(int[] a) {

int odd = 0;

int even = 0;

for (int i : a) {

if (i % 2 == 0) {

even += i;

} else {

odd += i;

}

}

return odd - even;

}

static int reverse(int n) {

int rev = 0;

while (n != 0) {

rev = rev \* 10 + n % 10;

n = n / 10;

}

return rev;

}

static int equilibrium(int[] a) {

if (a.length < 3) {

return -1;

}

int f = 0;

int m = 1;

int b = a.length - 1;

int sumf = a[f];

int sumb = a[b];

while (b - f > 2) {

if (sumf > sumb) {

b -= 1;

m = f + 1;

sumb += a[b];

} else {

m = b - 1;

f += 1;

sumf += a[f];

}

}

if (sumf == sumb) {

return m;

} else {

return -1;

}

}

static int largestAdjacentSum(int[] a) {

int len = a.length;

int sum = 0;

for (int i = 0; i < len - 1; i++) {

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

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

}

}

return sum;

}

static int checkConcatenatedSum(int n, int catlen) {

int sum = 0;

final int num = n;

while (n > 0) {

int tem = n % 10;

n = n / 10;

int tsum = tem;

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

tsum = tsum \* 10 + tem;

}

sum = sum + tsum;

}

if (num == sum) {

return 1;

}

return 0;

}

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

int len = a.length;

if (m != a[0] || n != a[len - 1]) {

return 0;

}

for (int i = 0; i < len - 1; i++) {

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

continue;

} else {

return 0;

}

}

return 1;

}

static int largestPrimeFactor(int n) {

if (n < 2) {

return 0;

}

if (n == 4) {

return 2;

}

if (n == 6) {

return 3;

}

int num = 0;

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

if (n % i == 0) {

int c = 0;

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

if (i % j == 0) {

c = 1;

break;

}

}

if (c == 0) {

num = i;

}

}

}

return num;

}

static int[] encodeNumber(int n) {

if (n <= 1) {

return null;

}

int[] arr = new int[n / 2];

int c = 0;

while (n > 1) {

if (n % 2 == 0) {

arr[c] = 2;

c++;

n /= 2;

} else if (n % 3 == 0) {

arr[c] = 3;

c++;

n /= 3;

} else if (n % 5 == 0) {

arr[c] = 5;

c++;

n /= 5;

} else {

for (int i = 6; i <= n; i++) {

if (n % i == 0) {

arr[c] = i;

c++;

n /= i;

break;

}

}

}

}

int[] arrN = new int[c];

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

arrN[i] = arr[i];

}

return arrN;

}

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

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

return 0;

}

int c = 0;

for (int i : a) {

if (i == pattern[c]) {

continue;

} else if (i == pattern[c + 1]) {

c++;

} else {

return 0;

}

}

return 1;

}

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

if (n <= 0) {

System.out.println(a);

} else {

int[] ar = new int[a.length];

int c = 0;

for (int i : a) {

final int k = n + 1; // k is used to adjast if the n value is odd

if (i < 1) {

ar[c] = i;

c++;

} else if (i % n < k / 2) {

ar[c] = (i / n) \* n;

c++;

} else {

ar[c] = (i / n + 1) \* n;

c++;

}

}

for (int i : ar) {

System.out.println(i);

}

System.out.println(ar);

}

}

static int isCubePowerful(int n) {

if (n < 1) {

return 0;

}

final int num = n;

int sum = 0;

while (n > 0) {

int tem = n % 10;

n /= 10;

sum += tem \* tem \* tem;

}

int r = (sum == num) ? 1 : 0;

return r;

}

static int decodeArray(int[] a) {

int sign = 1;

if (a[0] < 0) {

sign = -1;

}

int sum = 0;

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

int tsum = a[i] - a[i + 1];

if (tsum < 0) {

sum = sum \* 10 - tsum;

} else {

sum = sum \* 10 + tsum;

}

}

return sign \* sum;

}

static int isZeroPlentiful(int[] a) {

int count = 0;

int c0 = 0;

for (int i : a) {

if (i == 0) {

c0++;

} else {

if (c0 >= 4) {

count++;

c0 = 0;

} else {

c0 = 0;

}

}

}

int last0 = (c0 >= 4) ? 1 : 0;

return count + last0;

}

static int isDigitIncreasing(int n) {

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

int sum = 0;

int tem = 0;

while (n > sum) {

tem = tem \* 10 + i;

sum += tem;

}

if (n == sum) {

return 1;

}

}

return 0;

}

static int decodeArray1(int[] a) {

int len = a.length;

int sum = 0;

int sign = 1;

int c0 = 0;

if (a[0] == 1) {

sum = 0;

} else if (a[0] == 0) {

c0 = 1;

} else {

sign = -1;

}

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

if (a[i] == 0) {

c0++;

} else {

sum = sum \* 10 + c0;

c0 = 0;

}

}

return sum \* sign;

}

static int isPrimeHappy(int n) {

int sum = 0;

if (n <= 4) {

return 0;

}

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

int c = 0;

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

if (i % j == 0) {

c = 1;

break;

}

}

if (c == 0) {

sum += i;

}

}

int rt = (sum % n == 0) ? 1 : 0;

return rt;

}

static int[] encodeArray(int n) {

final int num = n;

if (n < 0) {

n = -1 \* n;

}

int[] ar = new int[n + 3];

int c = 0;

while (n > 0) {

int tem = n % 10;

n /= 10;

ar[c] = 1;

c++;

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

ar[c] = 0;

c++;

}

}

if (num == 0) {

int[] arn = {1};

return arn;

} else if (num < 0) {

ar[c] = -1;

c++;

}

int[] arN = new int[c];

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

arN[c - i] = ar[i - 1];

}

return arN;

}

static int isSystematicallyIncreasing(int[] a) {

int len = a.length;

int sum = 0;

int c = 0;

int n = a[len - 1];

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

sum += i;

}

if (sum != len) {

return 0;

}

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

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

if (a[c] != j) {

return 0;

}

c++;

}

}

return 1;

}

static int isFactorialPrime(int n) {

if (n == 2 || n == 3) {

return 1;

}

// eliminate 2& 3 for better use of for loop

int[] ar = new int[n];

int c = 0;

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

if (n % i == 0) {

return 0;

}

// prime number break contole

if (i == 2 || i == 3) {

ar[c] = i;

c++;

}

int b = 0;

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

if (i % j == 0) {

b = 1;

break;

}

}

if (b == 0) {

ar[c] = i;

c++;

}

}

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

int mul = 1;

for (int j = 2; j <= ar[i]; j++) {

mul \*= j;

}

if (n == mul + 1) {

return 1;

}

}

return 0;

}

static int largestDifferenceOfEvens(int[] a) {

int min = 0;

int max = 0;

int c = 0;

for (int i : a) {

if (i % 2 == 0) {

if (c == 0) {

min = i;

max = i;

} else if (i < min) {

min = i;

} else if (i > max) {

max = i;

}

c++;

}

}

if (c < 2) {

return -1;

} else {

return max - min;

}

}

static int isHodder(int n) {

if (n == 3) {

return 1;

} else if (n <= 2) {

return 0;

}

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

if (n % i == 0) {

return 0;

}

}

n = n + 1;

while (n > 2) {

n /= 2;

if (n % 2 != 0) {

return 0;

}

}

return 1;

}

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

if (a1.length != a2.length) {

return 0;

} else if (a1.length == 0 && a2.length == 0) {

return 1;

}

for (char i : a1) {

int c = 0;

for (char j : a2) {

if (j == i) {

c++;

}

}

if (c == 0) {

return 0;

} else if (c >= 1) {

for (char m : a1) {

if (i == m) {

c--;

}

}

if (c != 0) {

return 0;

}

}

c = 0;

}

return 1;

}

static int isPairedN(int[] a, int n) {

int l = a.length;

if (l <= 1 || n < 0) {

return 0;

}

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

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

if (i + j == n && a[i] + a[j] == n) {

return 1;

}

}

}

return 0;

}

public static void main(String[] arg) {

int[] a = {1, 2, 1, 2};

int[] a2 = {};

int n = 0;

// int m = 4;

String s = "isPairedN";

// System.out.println(method(s));

System.out.println(isOlympic(a));

}

}

static int isBrownian(int n){

// if(n<=0) return 0;

// int f =n\*n;

// int c =0;

// int b=n;

// while(b!=0){

// c++;

// b=b/10 ;

// }

// String checked = String.valueOf(f);

// if(checked.substring(checked.length()-c).equals(String.valueOf(n))) return 1;

// return 0;

// }

static int isOneHeavy(int[] a){

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

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

if(a[i]!=1 && a[i]!=0) return 0;

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

if(a[i+1]==1 && a[i+2] ==0) return 0;

}

return 1;

}