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;

}

return 1;

}

return 0;

}

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;

}

Printed upto here

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}));

}

}