1. **Happy Number**

class Solution {

private int getNext(int n) {

int totalSum = 0;

while (n > 0) {

int d = n % 10;

n = n / 10;

totalSum += d \* d;

}

return totalSum;

}

public boolean isHappy(int n) {

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

while (n != 1 && !seen.contains(n)) {

seen.add(n);

n = getNext(n);

}

return n == 1;

}

}

**2.UpCount**

public static int upCount(int[] arr, int num){

int partialSum=0;

int count=0;

int flag=1;

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

partialSum += arr[i];

if (partialSum>num) {

if (flag==1)

{

count++;

}

}

else if(count>0 && partialSum>num){

flag=0;

}

else if (count>0 && partialSum<num) {

flag=1;

}

}

return count;

}

**3.PrimeCount**

static boolean isPrime(int n) {

boolean flag = true;

if( n <=1)

flag = false;

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

if(n%i == 0)

flag = false;

break;

}

return flag;

}

static int primeCount(int start, int end) {

int count = 0;

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

if(isPrime(j))

count ++;

}

return count;

}

**4.LargestAdjacentSum**

package practice;

public class LargestAdjacentSum {

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;

}

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4};

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

}

**}**

**5.CatlenDigit**

public class CheckConcatenatedSum {

static int checkConcatenatedSum(int n, int catlen) {

int num = n;

int sum = 0;

int digit = 0;

while(num > 0) {

digit = num %10;

int catlenDigit = 0;

int multiple10 = 1;

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

catlenDigit +=(digit\* multiple10);

multiple10 \*=10;

}

sum += catlenDigit;

num /= 10;

}

if(sum ==n)

return 1;

else

return 0;

}

public static void main(String[] args) {

System.out.println(checkConcatenatedSum(13332, 4));

}

}

**5. IsSequencedArray**

package practice;

public class IsSequencedArray {

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;

}

public static void main(String[] args) {

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

System.out.println(isSequencedArray(arr, -5,-2));

}

}

**6.LargestPrime**

public class LargestPrimeFactor {

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;

}

public static void main(String[] args) {

System.out.println(largestPrimeFactor(10));

}

}

**7.Common Elements in two arrays**

public class CommonElemetns {

static int[] commonElements(int[] first, int[] second) {

if(first == null || second == null)

return null;

int count = 0;

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

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

if(first[i] == second[j]) {

count ++;

break;

}

}

}

int[] arr = new int[count];

if(count == 0)

return arr;

int index = 0;

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

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

if(first[i] == second[j]) {

arr[index] = first[i];

index ++;

break;

}

}

}

return arr;

}

public static void main(String[] args) {

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

}

}

**8. EncodeNumber**

public static void main(String args[])

{

System.out.println(Arrays.toString (PrimeFactors(1200)));

}

static int[] PrimeFactors(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;

}

9. **MatchPattern**

public class MatchPattern {

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;

}

public static void main(String[] args) {

int[] arr = {1,1,2,2,3,4,4,5,5,5,6,7};

int[] pattern = {1,2,3,4,5,6,7};

System.out.println(matchPattern(arr, pattern));

}

}

10. **IsCubePowerful**

public class IsCubePowerful {

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;

}

public static void main(String[] args) {

System.out.println(isCubePowerful(-81));

}

}

11. **Decode array**

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;

}

public static void main(String[] args) {

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

}

12.Is Zero Plentiful

public class IsZeroPlentiful {

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];

continue;

}

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

if(count < 4)

return 0;

sequence ++;

continue;

}

if(a[i] != 0) {

continue;

}

}

return sequence;

}

public static void main(String[] args) {

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

}

}

12. **isDigitIncreasing**

static int isDigitIncreasing(int n) {

if(n < 10)

return 1;

int firstDigit = 0;

int count = 0;

int num = n;

int digit = 0;

while(num > 0) {

digit = num%10;

count ++;

firstDigit = digit;

num /=10;

}

int multiple10 = 10;

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

int sumDigit = i;

int counter = count;

int sum = 0;

while(counter > 0) {

sum +=sumDigit;

sumDigit \*= multiple10;

sumDigit += i;

counter --;

}

if(sum == n)

return 1;

}

return 0;

}

public static void main(String[] args) {

System.out.println(isDigitIncreasing(7404));

}

**13.IsPrimeHappy**

public class IsPrimeHappy {

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;

}

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;

}

public static void main(String[] args) {

System.out.println(isPrimeHappy(2));

}

}

**14.IsFactorialPrime**

public class IsFactorialPrime {

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;

}

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;

}

static int factorial(int n) {

int factorial = 1;

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

factorial \*= i;

}

return factorial;

}

public static void main(String[] args) {

System.out.println(isFactorialPrime(8));

}

**15.ClosestFibonacci**

public class ClosestFibonacci {

static int closestFibonacci(int n) {

if(n < 1)

return 0;

int previous = 1;

int next = 1;

while(next < n) {

int temp = next;

next = previous + next;

previous = temp;

}

if(next > n)

return previous;

return next;

}

public static void main(String[] args) {

System.out.println(closestFibonacci(100));

}

**15.isVesuvian**

public class IsVesuvian {

static int isVesuvian(int n) {

int count = 0;

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

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

int sum = i + j;

if(sum == n && isSquare(i) && isSquare(j)) {

count ++;

if(count > 1)

return 1;

}

}

}

return 0;

}

static boolean isSquare(int n) {

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

if(n%i == 0) {

int x = n/i;

if(x == i)

return true;

}

}

return false;

}

public static void main(String[] args) {

System.out.println(isVesuvian(85));

}

}

**16.IsTraingular**

static int isTriangular(int n) {

int sum=0;

int i;

int flag=0;

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

{

sum=sum+i;

if (sum==n)

{

flag=1;

}

}

if(flag==1)

return 1;

return 0;

}

**17. IsMarcurial**

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;

}

public static void main(String[] args) {

int[] arr = {8,2,1,1,18,3,5};

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

}

**18.IsMartian**

public class IsMartian {

static int isMartian(int[] a) {

int countOnes = 0;

int countTwos = 0;

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

{

if(a[i] == 1)

countOnes ++;

if(a[i] == 2)

countTwos ++;

}

if(countOnes <= countTwos)

return 0;

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

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

return 0;

}

return 1;

}

public static void main(String[] args) {

System.out.println(isMartian(new int[] {2}));

}

}

**19. IsPairedN**

public class IsPairedN {

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

int length = a.length;

int maxSum = (length-1) + (length -2);

if(n < 0 || n > maxSum || length < 2)

return 0;

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

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

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

return 1;

}

}

return 0;

}

public static void main(String[] args) {

int[] arr = {1, 4, 1};

System.out.println(isPairedN(arr, 5));

}

}

**20. IsNPrimeable**

public class IsNPrimeable {

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;

}

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) {

int[] arr = {};

System.out.println(isNPrimeable(arr, 2));

}

**21. is121Array**

public class Is121Array {

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;

}

}

**22. PairwiseSum**

public class PairwiseSum {

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

int length = a.length;

if(length%2 != 0 || length == 0)

return null;

int[] result = new int[length/2];

int index = 0;

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

result[index] = a[i] + a[j];

index ++;

}

return result;

}

public static void main(String[] args) {

System.out.println(Arrays.toString(pairwiseSum(new int[] {2, 1, 18})));

}

**23. IsSquare**

public class IsSquare {

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;

}

public static void main(String[] args) {

IsSquare is = new IsSquare();

System.out.println(is.isSquare(1));

}

**24.isComplete**

public class IsComplete {

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;

}

public static void main(String[] args) {

System.out.println(isComplete(new int[] {3, 2, 1, 1, 5, 6}));

}

}

**25.Loopsum**

public class LoopSum {

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;

}

public static void main(String[] args) {

System.out.println(loopSum(new int[] {1, 4, 5, 6}, 4));

}

**26. HasNValues**

public class HasNValues {

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;

}

public static void main(String[] args) {

System.out.println(hasNValues(new int[]{1, 2, 3, 4, 5, 6, 7, 8 ,9, 10}, 10));

}

}

**27. SameNumberOfFactors**

public class SameNumberOfFactors {

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;

}

public static void main(String[] args) {

System.out.println(sameNumberOfFactors(0, 0));

}

}

**28. Eval**

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;

}

public static void main(String[] args) {

System.out.println(eval(10.0, new int[] {0, 1} ));

}

**29. isConsecutiveFactored**

static int isConsecutiveFactored(int n) {

int previous = 1;

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

if(n%i == 0)

{

if(previous >1)

{

if(i == (previous+1))

return 1;

}

previous = i;

}

}

return 0;

}

public static void main(String[] args) {

System.out.println(isConsecutiveFactored(90));

}

**29. IsTwinPrime**

public class IsTwinPrime {

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;

}

public static void main(String[] args) {

System.out.println(isTwinPrime(53));

}

}

**30. IsZeroBalanced**

public class IsZeroBalanced {

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;

}

public static void main(String[] args) {

System.out.println(isZeroBalanced(new int[] {3,-3,-3}));

}

**31. percentDivisibleBy3**

static float percentDivisibleBy3(int[] a) {

float count = 0.0f;

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

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

count ++;

}

return count/a.length\*100;

}

**32. IsSymmetric**

public class IsSymmetric {

static int isSymmetric(int[] a) {

if(a.length%2 == 1) {

int mid = a.length/2;

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

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

return 0;

}

}else {

int mid1 = a.length/2; // index 3

int mid2 = a.length/2-1; // index 2

for(int i= mid1+1, j= mid2-1; i < a.length; i++, j--) {

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

return 0;

}

}

return 1;

}

public static void main(String[] args) {

System.out.println(isSymmetric(new int[] {4, 5, 6, 6, 5, 4}));

}

**33. distinctIntegers**

static int distinctIntegers(int[] a) {

int count = 0;

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

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

{

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

continue outer;

}

count ++;

}

return count;

}

public static void main(String[] args) {

System.out.println(distinctIntegers(new int[] {5,5,5,5}));

}