import java.util.\*;

public class Delete {

public static void main(String[] args) {

String s = "eavdivOs";

System.out.println(delete(s));

}

public static String delete(String s) {

Set<Character> set = new HashSet<>(Arrays.asList('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'));

char[] ch = s.toCharArray();

StringBuilder sb = new StringBuilder();

for (char c : ch) {

if (set.contains(c)) {

continue;

}

sb.append(c);

}

return sb.toString();

}

}

import java.util.Arrays;

public class FourSum {

public static void main(String[] args) {

int[] nums = {1, 0, -1, 0, -2, 2};

int target = 0;

System.out.println(fourSum(nums, target));

}

public static int fourSum(int[] nums, int target) {

int res = 0;

if (nums.length < 4) return res;

Arrays.sort(nums);

for (int i = 0; i < nums.length - 3; i++) {

if (i > 0 && nums[i] == nums[i - 1]) continue;

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

if (j > i + 1 && nums[j] == nums[j - 1]) continue;

int low = j + 1, high = nums.length - 1;

while (low < high) {

int sum = nums[i] + nums[j] + nums[low] + nums[high];

if (sum == target) {

res++;

while (low < high && nums[low] == nums[low + 1]) low++;

while (low < high && nums[high] == nums[high - 1]) high--;

high--;

low++;

} else if (sum < target) {

low++;

} else high--;

}

}

}

return res;

}

}

public String direction(int[] x, int[] y) {

if (x[0] == y[0] && x[1] == y[1]) return "here";

if (x[0] == y[0] && x[1] < y[1]) return "N";

if (x[0] == y[0] && x[1] > y[1]) return "S";

if (x[0] < y[0] && x[1] == y[1]) return "E";

if (x[0] > y[0] && x[1] == y[1]) return "W";

if (x[0] < y[0] && x[1] < y[1]) return "NE";

if (x[0] < y[0] && x[1] > y[1]) return "SE";

if (x[0] > y[0] && x[1] > y[1]) return "SW";

if (x[0] > y[0] && x[1] < y[1]) return "NW";

return "-1";

}

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class Reverse {

public static void main(String[] args) {

System.out.println(NPalindrome(195));

}

static int time = 0;

static int sum = 0;

public static List<Integer> NPalindrome(int x) {

int rev = reverse(x);

if (rev != x) {

sum = x + rev;

time++;

NPalindrome(sum);

}

return new ArrayList<>(Arrays.asList(time, sum));

}

public static int reverse(int x) {

int y = x;

int res = 0;

while(y != 0) {

res = res \* 10 + y % 10;

y /= 10;

}

return res;

}

}