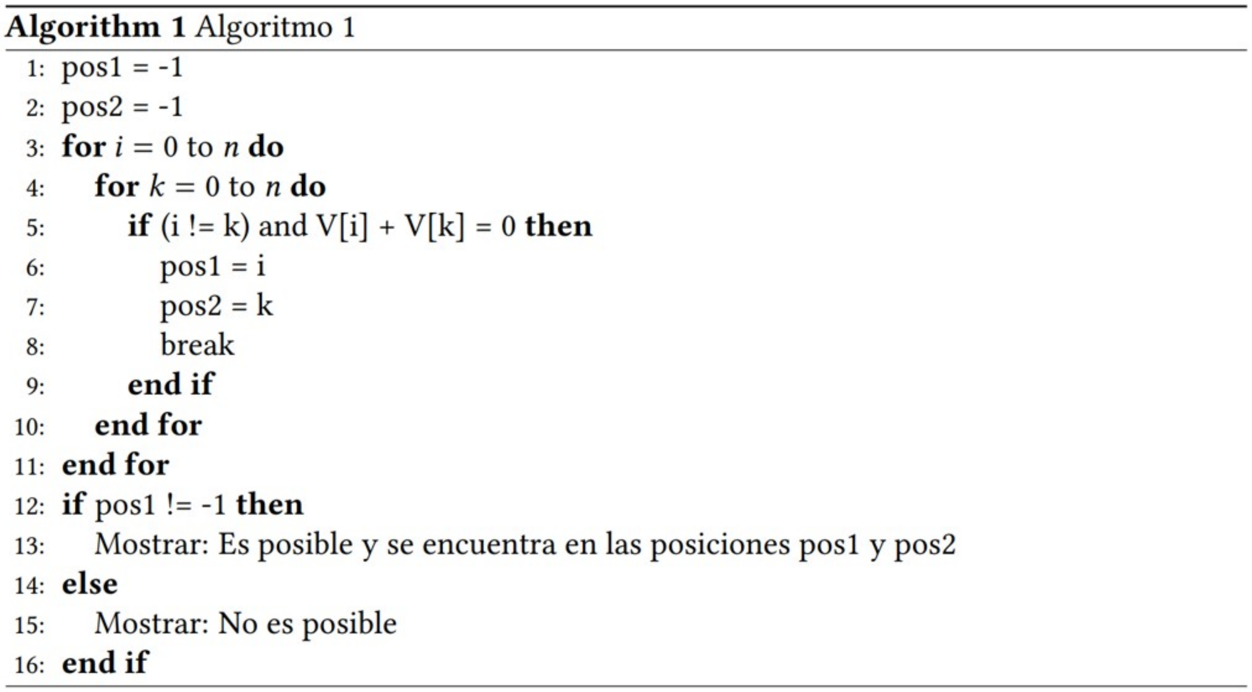
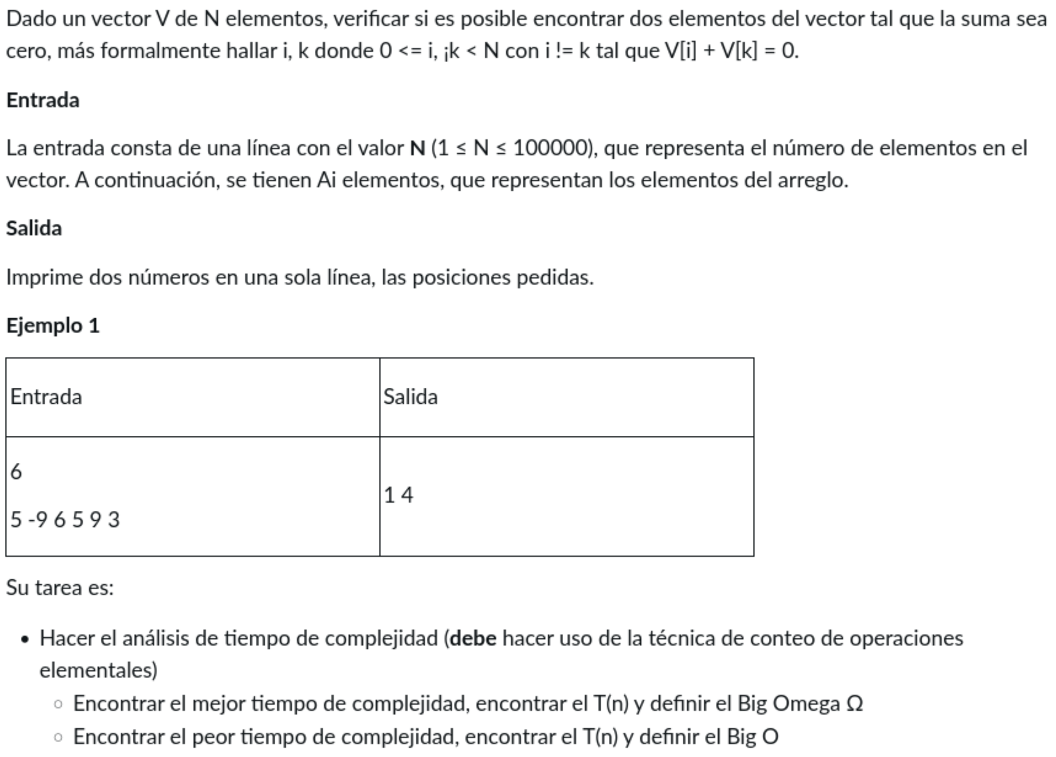
Jesus Leonardo Escobar Herrera Laboratorio 2 Algoritmos 1

Actividad #1 – Encontrar el tiempo de complejidad



Algorithm 1 Algoritmo 1

```
 1: pos1 = -1
 2: pos2 = -1
 3: for $i = 0$ to $n$ do
 4:     for $k = 0$ to $n$ do
 5:         if (i != k) and V[i] + V[k] = 0 then
 6:             pos1 = i
 7:             pos2 = k
 8:             break
 9:         end if
10:     end for
11: end for
12: if pos1 != -1 then
13:     Mostrar: Es posible y se encuentra en las posiciones pos1 y pos2
14: else
15:     Mostrar: No es posible
16: end if
```



Dado un vector V de N elementos, verificar si es posible encontrar dos elementos del vector tal que la suma sea cero, más formalmente hallar i, k donde 0 <= i, ¡k < N con i != k tal que V[i] + V[k] = 0.

Entrada

La entrada consta de una línea con el valor **N** (1 ≤ N ≤ 100000), que representa el número de elementos en el vector. A continuación, se tienen Ai elementos, que representan los elementos del arreglo.

Salida

Imprime dos números en una sola línea, las posiciones pedidas.

Ejemplo 1

Entrada	Salida
6 5 -9 6 5 9 3	1 4

Su tarea es:

- Hacer el análisis de tiempo de complejidad (**debe** hacer uso de la técnica de conteo de operaciones elementales)
    - Encontrar el mejor tiempo de complejidad, encontrar el T(n) y definir el Big Omega Ω
    - Encontrar el peor tiempo de complejidad, encontrar el T(n) y definir el Big O

```
1: pos1 = -1     1OE
2: pos2 = -1     1OE
3: for i = 0 to n do      (Analisis 1)
4:     for k = 0 to n do (Analisis 2)
5:         if (i != k) and V[i] + V[k] = 0 then
6:             pos1 = i
7:             pos2 = k
8:             break
9:         end if
10:    end for
11: end for
12: if pos1 != -1 then      1OE
13:     Mostrar: Es posible y se encuentra en las posiciones pos1 y pos2     1OE
14: else
15:     Mostrar: No es posible     1OE
16: end if
```

Operaciones fuera de los “for” = 5 OE

$$T(n) = T(inicializacion) + T(comparacion) + N * \begin{pmatrix} T(instrucciones) + \\ T(comparacion) + \\ T(incremento\ o\ decremento) \end{pmatrix}$$

Propiedades

1. $\sum_{i=a}^{n} k = (n-a+1)k$

2. $\sum_{i=a}^{n} [f(i)+g(i)] = \sum_{i=a}^{n} f(i) + \sum_{i=a}^{n} g(i)$

3. $\sum_{i=a}^{n} c\, f(i) = c \sum_{i=a}^{n} f(i)$

4. $\sum_{i=1}^{n} [f(i) - f(i-1)] = f(n) - f(0)$

(Calculo Integral Conamat) Primera edicion. Pag 4

Analisis 1

```
3: for i = 0 to n do      (Analisis 1)
4:     for k = 0 to n do (Analisis 2)
5:         if (i != k) and V[i] + V[k] = 0 then
6:             pos1 = i
7:             pos2 = k
8:             break
9:         end if
10:    end for
11: end for
```

t inicializacion = 1

t comparacion = 1

t instrucciones = Analisis 2 (ver Analisis antes de seguir)

t incremento = 2

$$T(n)=1+1+\sum_{i=0}^{n-1}((4n+2)+1+2)$$

$$T(n)=2+\sum_{i=0}^{n-1}((4n+2)+1+2)$$

$$T(n)=2+(n)(4n+5)=4n^2+5n+2$$

$$4n^2+5n+2+5(\mathit{Operaciones\ fuera\ del}\ \text{for})=4n^2+5n+2+7$$

Por lo tanto:

- **Big Ω(1) – Mejor caso**
- **Big O(n²) – En el peor caso**

Analisis 2

```
4:   for k = 0 to n do (Analisis 2)
5:       if (i != k) and V[i] + V[k] = 0 then 3OE
6:           pos1 = I 1OE
7:           pos2 = k 1OE
8:           break
9:       end if
```

t inicializacion = 1

t comparacion = 1

t instrucciones = 5 OE

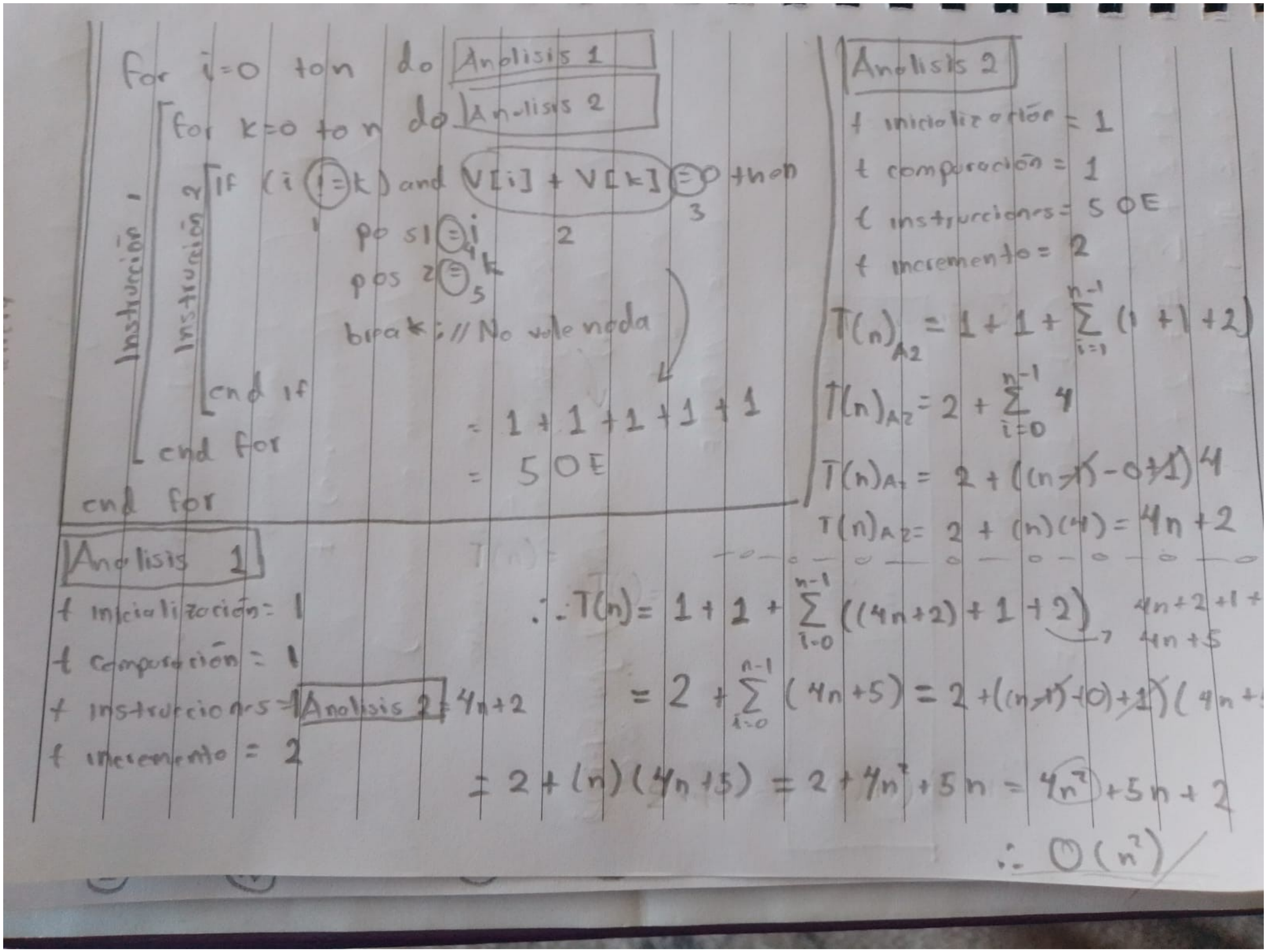
t incremento = 2

$$T(n)=1+1+\sum_{i=0}^{n-1}(1+1+2)$$

$$T(n)=2+\sum_{i=0}^{n-1}(4)$$

$$T(n)=2+((n-1)-0+1)4$$

$$T(n)=4n+2$$



```
for i=0 to n do                                   [Análisis 1]
    for k=0 to n do                               [Análisis 2]
        if (i != k) and V[i] + V[k] == 0 then
            pos1 = i
            pos2 = k
            break; // No vale nada
        end if
    end for
end for
```

Instrucción 1

Instrucción 2

$= 1 + 1 + 1 + 1 + 1$

$= 5$ OE

$\boxed{\text{Análisis 2}}$

t inicialización = 1

t comparación = 1

t instrucciones = 5 OE

t incremento = 2

$T(n)_{A2} = 1 + 1 + \sum_{i=1}^{n-1} (1 + 1 + 2)$

$T(n)_{A2} = 2 + \sum_{i=0}^{n-1} 4$

$T(n)_{A2} = 2 + ((n-\cancel{1}) - 0 + \cancel{1})4$

$T(n)_{A2} = 2 + (n)(4) = 4n + 2$

$\boxed{\text{Análisis 1}}$

t inicialización = 1

t comparación = 1

t instrucciones = $\boxed{\text{Análisis 2}}$ = $4n+2$

t incremento = 2

$\therefore T(n) = 1 + 1 + \sum_{i=0}^{n-1} ((4n+2) + 1 + 2)$ $\quad 4n+2+1+ \to 4n+5$

$= 2 + \sum_{i=0}^{n-1} (4n+5) = 2 + ((n-\cancel{1}) - 0 + \cancel{1})(4n+$

$= 2 + (n)(4n+5) = 2 + 4n^2 + 5n = 4n^2 + 5n + 2$

$\therefore O(n^2)$

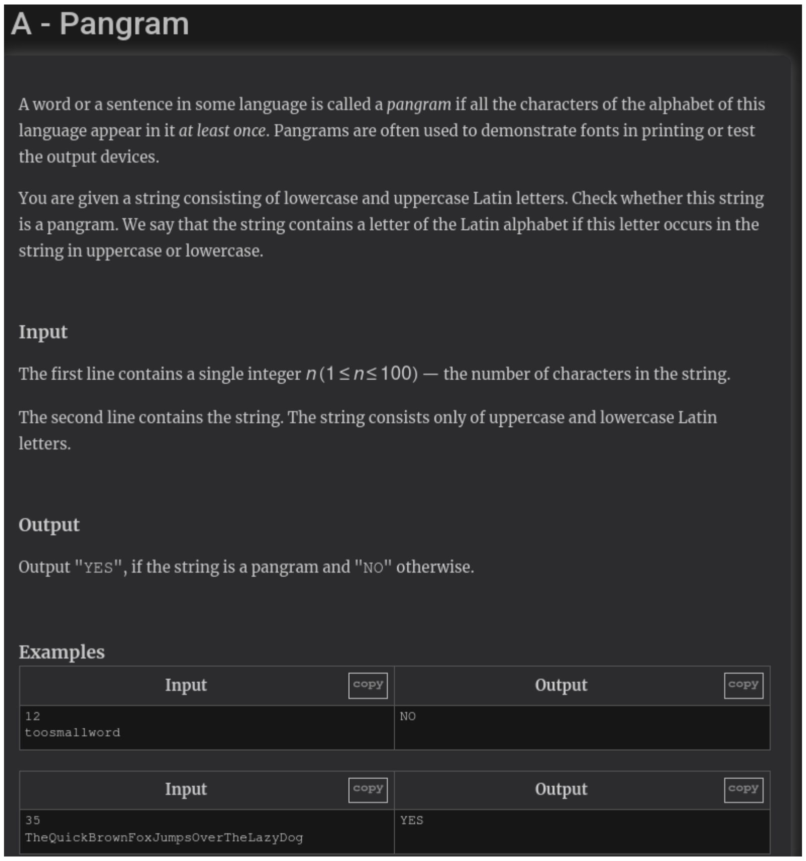
Jesus Leonardo Escobar Herrera Laboratorio 2 Algoritmos 1

Actividad #2 – Resolucion de problema

Resolver los ejercicios A y B del contest llamado “Contest semana 2” del grupo “AlgoritmosI_Seccion”(A, B o C)

- Analizar el problema y explicar la idea de la solución aplicada.
- Resolver el problema ya sea en *Java, C++ o Python*y subirlo al contest. Asegurase que le de AC como veredicto.
- Hacer el análisis de tiempo de complejidad (**debe** hacer uso de la técnica de conteo de operaciones elementales)
    - Encontrar el mejor tiempo de complejidad, encontrar el T(n) y definir el Big Omega Ω
    - Encontrar el peor tiempo de complejidad, encontrar el T(n) y definir el Big O

Ejercicio 1



A - Pangram

A word or a sentence in some language is called a *pangram* if all the characters of the alphabet of this language appear in it *at least once*. Pangrams are often used to demonstrate fonts in printing or test the output devices.

You are given a string consisting of lowercase and uppercase Latin letters. Check whether this string is a pangram. We say that the string contains a letter of the Latin alphabet if this letter occurs in the string in uppercase or lowercase.

Input

The first line contains a single integer n ($1 \le n \le 100$) — the number of characters in the string.

The second line contains the string. The string consists only of uppercase and lowercase Latin letters.

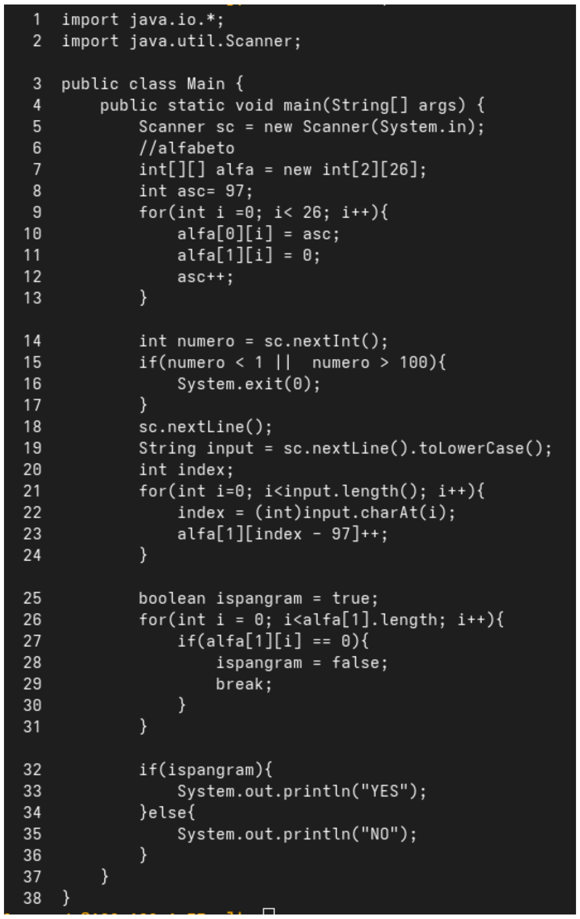
Output

Output "YES", if the string is a pangram and "NO" otherwise.

Examples

Input	Output
12 toosmallword	NO

Input	Output
35 TheQuickBrownFoxJumpsOverTheLazyDog	YES



```java
import java.io.*;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        //alfabeto
        int[][] alfa = new int[2][26];
        int asc= 97;
        for(int i =0; i< 26; i++){
            alfa[0][i] = asc;
            alfa[1][i] = 0;
            asc++;
        }

        int numero = sc.nextInt();
        if(numero < 1 ||  numero > 100){
            System.exit(0);
        }
        sc.nextLine();
        String input = sc.nextLine().toLowerCase();
        int index;
        for(int i=0; i<input.length(); i++){
            index = (int)input.charAt(i);
            alfa[1][index - 97]++;
        }

        boolean ispangram = true;
        for(int i = 0; i<alfa[1].length; i++){
            if(alfa[1][i] == 0){
                ispangram = false;
                break;
            }
        }

        if(ispangram){
            System.out.println("YES");
        }else{
            System.out.println("NO");
        }
    }
}
```

```
import java.io.*;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        //alfabeto
        int[][] alfa = new int[2][26]; 1OE

        int asc= 97; 1OE
        for(int i =0; i< 26; i++){ (1(iniciacion)+1(comprobacion)+2(incremento)) (n)
            alfa[0][i] = asc; 2OE 1(acceso al array), 1(asignacion)
            alfa[1][i] = 0; 2OE
            asc++; 2OE
        }

        int numero = sc.nextInt(); 1OE
        if(numero < 1 || numero > 100){ 2OE
            System.exit(0); 1OE
        }

        sc.nextLine();
        String input = sc.nextLine().toLowerCase(); 3OE
        int index;
        for(int i=0; i<input.length(); i++){ (1(iniciacion)+1(comprobacion)+2(incremento)) (n)
            index = (int)input.charAt(i); 1(asignacion) + 1(charAt) = 2OE (no se si el casteo cuente
como una o es propia del lenguaje????)
            alfa[1][index – 97]++; 1(acceso al array) + 1(operacion resta) + 2(incremento) = 4OE
        }

        boolean ispangram = true; 1OE
        for(int i = 0; i<alfa[1].length; i++){ (1(iniciacion)+1(comprobacion)+2(incremento)) (n)
            if(alfa[1][i] == 0){ 2OE
                ispangram = false; 1OE
                break;
            }
        }

        if(ispangram){ 1OE
            System.out.println("YES"); 1OE
        }else{
            System.out.println("NO");1OE
        }
    }
}
```

Operaciones fuera del “for” = 13 OE

Primer “for” =

$$T(n)=1+1+\sum_{i=0}^{n-1}(6)=2+((n-1)-0+1)(6)=6n+2$$

Segundo “for” =

$$T(n)=1+1+\sum_{i=0}^{n-1}(6)=2+((n-1)-0+1)(6)=6n+2$$

Tercer “for” =

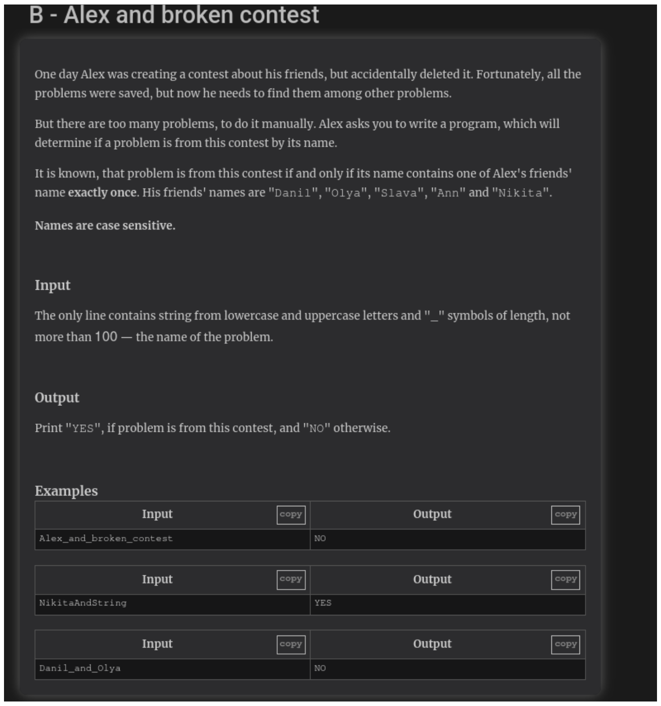
$$T(n)=1+1+\sum_{i=0}^{n-1}(3)=2+((n-1)-0+1)(3)=3n+2$$

Total = 13 + 6n+2+6n+2+3n+2 = 9n+19

- **Big Ω(1) – Mejor caso**
- **Big O(n) – En el peor caso**

Jesus Leonardo Escobar Herrera Laboratorio 2 Algoritmos 1

Ejercicio 2



B - Alex and broken contest

One day Alex was creating a contest about his friends, but accidentally deleted it. Fortunately, all the problems were saved, but now he needs to find them among other problems.

But there are too many problems, to do it manually. Alex asks you to write a program, which will determine if a problem is from this contest by its name.

It is known, that problem is from this contest if and only if its name contains one of Alex's friends' name **exactly once**. His friends' names are "Danil", "Olya", "Slava", "Ann" and "Nikita".

Names are case sensitive.

Input

The only line contains string from lowercase and uppercase letters and "_" symbols of length, not more than 100 — the name of the problem.

Output

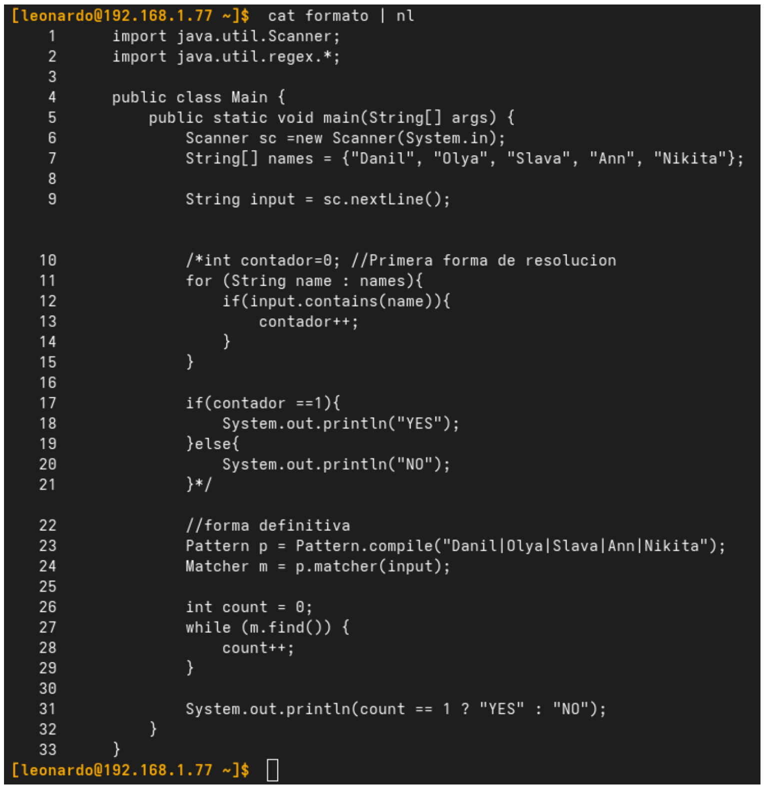
Print "YES", if problem is from this contest, and "NO" otherwise.

Examples

Input	Output
Alex_and_broken_contest	NO

Input	Output
NikitaAndString	YES

Input	Output
Danil_and_Olya	NO



```
[leonardo@192.168.1.77 ~]$  cat formato | nl
     import java.util.Scanner;
     import java.util.regex.*;

     public class Main {
         public static void main(String[] args) {
             Scanner sc =new Scanner(System.in);
             String[] names = {"Danil", "Olya", "Slava", "Ann", "Nikita"};

             String input = sc.nextLine();


             /*int contador=0; //Primera forma de resolucion
             for (String name : names){
                 if(input.contains(name)){
                     contador++;
                 }
             }

             if(contador ==1){
                 System.out.println("YES");
             }else{
                 System.out.println("NO");
             }*/

             //forma definitiva
             Pattern p = Pattern.compile("Danil|Olya|Slava|Ann|Nikita");
             Matcher m = p.matcher(input);

             int count = 0;
             while (m.find()) {
                 count++;
             }

             System.out.println(count == 1 ? "YES" : "NO");
         }
     }
[leonardo@192.168.1.77 ~]$
```

```
 1      import java.util.Scanner;
 2      import java.util.regex.*;
 3
 4      public class Main {
 5          public static void main(String[] args) {
 6              Scanner sc =new Scanner(System.in);
 7              String[] names = {"Danil", "Olya", "Slava", "Ann", "Nikita"}; 1OE
 8
 9              String input = sc.nextLine(); 1OE


10              /*int contador=0; //Primera forma de resolucion
11              for (String name : names){
12                  if(input.contains(name)){
13                      contador++;
14                  }
15              }
16
17              if(contador ==1){
18                  System.out.println("YES");
19              }else{
20                  System.out.println("NO");
21              }*/

22                  //forma definitiva (https://www.w3schools.com/java/java_regex.asp)
23              Pattern p = Pattern.compile("Danil|Olya|Slava|Ann|Nikita"); 2OE
24              Matcher m = p.matcher(input); 2OE //Sin embargo aqui solo contaremos 1, por que la
inicializacion la contaremos en la linea 27
25
26              int count = 0; 1OE
27              while (m.find()) { (1(iniciacion)+1(comprobacion)+2(incremento)) (n)
28                  count++; 2OE
29              }
30
31              System.out.println(count == 1 ? "YES" : "NO"); 3OE
32          }
33      }
```

Operaciones fuera del “for” = 8 OE

Ciclo while =

$$T(n)=1+1+\sum_{i=0}^{n-1}(2+1+1)=2+((n-1)-0+1)(4)=4n+2$$

Total = 8+4n+2 = 4n+10

- **Big Ω(1) – Mejor caso**
- **Big O(n) – En el peor caso**