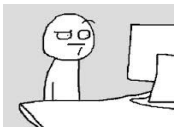


Universidad del País Vasco Euskal Herriko Unibertsitatea

Prog. Básica - Laboratorio 9
Matrices y estructuras de datos

Nombre:_____**Fecha**:_____

AVISOS:

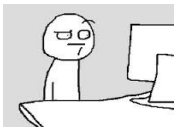
1. Cuando uno de los ejercicios que se proponen carece de programa de prueba o de plantilla, significa que lo tienes que hacer desde cero. Además, el hecho de que se proporcionen algunos casos de prueba no significa que no pueda faltar alguno. ¡Tienes que añadir los casos de prueba que falten!

2. Tienes que subir a eGela los ficheros fuente (.adb) comprimidos en un fichero (.zip) que deberá ajustarse a las reglas de nombrado exigidas hasta ahora (por ejemplo, EJauregi_lab7.zip).

3. Se presupone que los ejercicios son correctos, es decir, que no tienen errores de compilación y que funcionan correctamente. Esto significa que las soluciones de los ejercicios no puntúan nada por ser correctas, pero penalizan si tienen errores o no se ajustan a lo que se pide. Una vez que la solución es correcta, lo que se evalúa (lo que puntúa) son:

- **Los casos de prueba: ¿Se han contemplado todos los casos de prueba, desde los generales a los más críticos? Se dan algunos, pero faltan muchos otros.**
- **La eficiencia: ¿Se utilizan los "chivatos" cuando hace falta? ¿No hay asignaciones innecesarias? ¿No hay condicionales que no deberían ejecutarse?, ¿Se definen solamente los parámetros o variables necesarios?, etcétera.**
- **La claridad: ¿El código está tabulado? ¿Los nombres de las variables ayudan a entender el código? ¿Hay un único *return* al final de la función?, etcétera.**
- **Quien quiera hacer los ejercicios en PYTHON puede utilizar la siguiente dirección para implementar los ejercicios:**

https://py3.codeskulptor.org



1. Matriz_Toeplitz

Implementa en Ada un subprograma con nombre *matriz_toeplitz* que, dado un vector, cree una matriz de Toeplitz utilizando dicho vector. El vector es de un tamaño indeterminado y la matriz es una matriz cuadrada de tantas filas y columnas como elementos tiene el vector. La matriz toeplitz es aquélla en la que los elementos de sus diagonales (de izquierda a derecha) son iguales, es decir, $\forall a_{i,j} \in M \; a_{i,j} = a_{i+1,j+1}$, y gráficamente:

$$\begin{matrix} a_1 & b_2 & b_3 & b_4 \\ a_2 & a_1 & b_2 & b_3 \\ a_3 & a_2 & a_1 & b_2 \\ a_4 & a_3 & a_2 & a_1 \end{matrix}$$

Como idea, el vector recibido puede ser la primera fila de la matriz, y las demás filas ser una rotación de la anterior.

Plantillas ofrecidas:

matriz_toeplitz.adb	Para hacer el subprograma en Ada
prueba_matriz_toeplitz.adb	Para completar y probar el subprograma en Ada

Fichero adicional: datos.ads, escribir_matriz.adb (No hay que modificarlo).

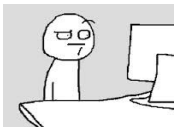
2. Lluvia

Se pide implementar un subprograma, que obtenga la provincia y el mes en el que más ha llovido.

Plantillas ofrecidas:

calcular_max_pico_de_lluvia.adb	Para hacer el subprograma en Ada
prueba_calcular_max_pico_de_lluvia.adb	Para completar y probar el subprograma en Ada

Fichero adicional: lluvia.ads (No hay que modificarlo).



3. Copia de exámenes

a) **se han copiado:**

De cara a evitar las copias de exámenes, una profesora ha diseñado un método que se basa en la frecuencia de aparición de las palabras del examen. El método se basa en codificar un examen como la lista de las palabras que aparecen en él, junto con el número de apariciones de cada palabra.
La idea básica para detectar copias es que los programas copiados tengan el mismo número de apariciones de variables (los nombres pueden no coincidir, ya que es un método habitual de copia el cambiar únicamente los nombres de las variables). Por ejemplo, los procedimientos a y b son sospechosos de copia, pero no si los comparamos con el c:

```
procedure a is
  x1, x2: integer;
begin
  if x1 > x2
  then v4 := 0;
  end if;
end a;
```

```
procedure b is
  y1, y2: integer;
begin
  if y1 > y2
  then indice := 0;
  end if;
end b;
```

```
procedure c is
  z1, z2, z3: integer;
begin
  if z1 > z2 + z3
  then z3 := Z3 + 1;
  end if;
end c;
```

Para codificar un examen, la profesora ha cogido las palabras que aparecen en él y las ha metido en una estructura de datos, contando por cada palabra cuántas veces aparece. En este proceso las palabras se guardan ordenadas alfabéticamente y además no se cuentan las palabras reservadas (if, while, for, procedure, ...). Por ejemplo, ésta podría ser la representación de 4 exámenes diferentes:

examen 1		examen 2		examen 3		examen 4	
5		5		5		6	
i	12	elem	7	elem	7	cont	12
indice	15	indice2	15	indice2	15	cont2	3
var3	7	j	9	j	12	j	6
var5	9	x	12	var3	9	w	24
var7	12	z	12	var19	9	z1	17
						z5	13

En este ejemplo, los exámenes 1 y 2 son sospechosos de copia, al contener el mismo número de variables y además, las frecuencias de aparición coinciden (15, 12, 12, 9, 7). Por su parte, los exámenes 1 y el 3 no son sospechosos de copia, porque las frecuencias son diferentes ((15, 12, 12, 9, 7) y (15, 12, 9, 9, 7)). El examen 4 no se considera copia de los demás porque su número de palabras es diferente de los otros tres.

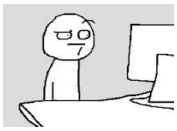
```
function se_han_copiado(Ex1, Ex2 : in Examen) return Boolean

-- precondición: Ex1 y Ex2 están ordenadas alfabéticamente

-- postcondición: El resultado es true si el número de palabras de

--            Ex1 y Ex2 es el mismo y además las palabras en su

--            conjunto coinciden en el número de apariciones
```



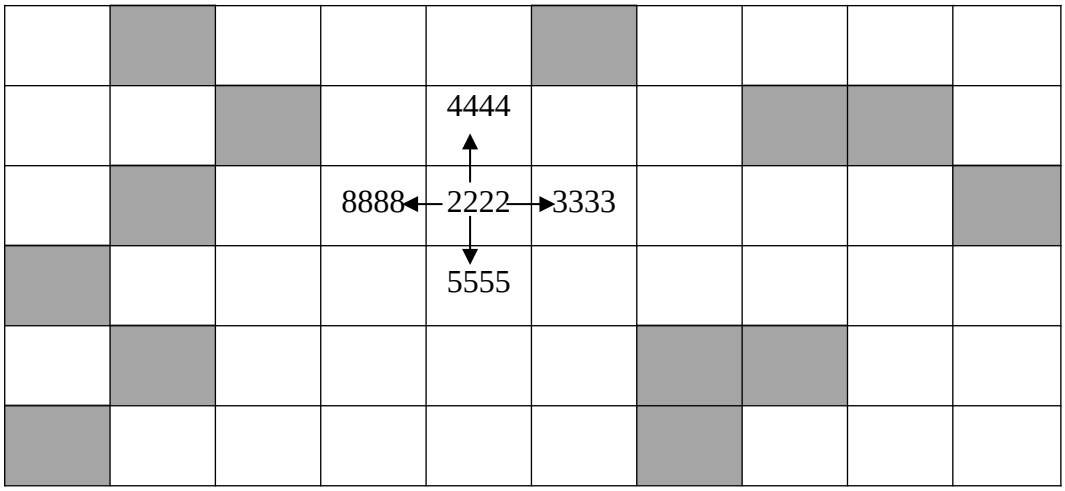
Plantillas ofrecidas:

posicion.adb	Para hacer el subprograma en Ada
se_han_copiado.adb	Para hacer el subprograma en Ada
prueba_se_han_copiado.adb	Para completar y probar el subprograma Ada

Fichero adicional: copia_examenes.ads

b) **escribir_sospechosos:**

Se tiene una matriz con los datos del examen de programación. Cada elemento de la matriz representa dónde se sentó cada alumno. Por cada alumno se tiene su número de identificación junto con la codificación del examen que ha realizado.



Se quiere sacar una lista de las parejas de alumnos que son sospechosos de copia. Sabemos que un alumno puede copiar a los alumnos que se encuentran delante, detrás, a su derecha o a su izquierda.

```
procedure escribir_sospechosos(Aula1: in Aula)
-- precondición: Aula1 contiene los datos de los alumnos que han
--               realizado un examen
-- postcondición: se han escrito en la pantalla las parejas de
--     alumnos sospechosos de haber copiado. En la lista no deberá
--     haber repeticiones de parejas, es decir, si aparece la pareja
--     (X, Y), no se deberá escribir la pareja (Y, X)
```

Plantillas ofrecidas:

se_han_copiado.adb	Para hacer el subprograma en Ada
escribir_sospechosos.adb	Para hacer el subprograma en Ada
prueba_escribir_sospechosos.adb	Para completar y probar el subprograma Ada

Fichero adicional: rellenar_aula_1.adb, copia_examenes.ads, escribir_aula_act.adb