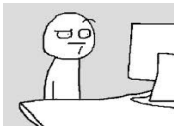


Universidad del País Vasco Euskal Herriko Unibertsitatea

Prog. Básica - Laboratorio 7
Vectores y matrices

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

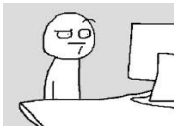
1. Búsqueda en un vector

Crear un subprograma que, dado un valor entero y un vector de N enteros (completo, o sea, con N enteros) cuyos valores pueden no estar ordenados, diga si el entero está o no en el vector.

Plantillas ofrecidas:

esta_en_vector.adb	Para hacer el subprograma en Ada
prueba_esta_en_vector.adb	Para completar y probar el subprograma en Ada

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



2. Búsqueda en un vector ordenado

Crear un subprograma que, dado un valor entero y un vector de N enteros (completo, o sea, con N enteros) cuyos valores están ordenados ascendentemente, diga si el entero está o no en el vector.

NOTA: Es imprescindible que la solución sea eficiente. En concreto, no se seguirá buscando el número cuando se sabe objetivamente (porque éste está ordenado) que no está en el vector.

Plantillas ofrecidas:

esta_en_vector_ordenado.adb	Para hacer el subprograma en Ada
prueba_esta_en_vector_ordenado.adb	Para completar y probar el subprograma Ada

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

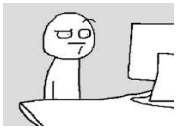
3. Búsqueda de la posición en un vector

Crear un subprograma que, dado un valor entero y un vector de N enteros (completo, o sea, con N enteros) cuyos valores pueden no estar ordenados, diga en qué posición se encuentra el entero en el vector. Si dicho entero estuviera repetido, bastará con devolver la posición de una cualquiera de sus apariciones; y en caso de no encontrarse en el vector, se devolverá el valor -1.

Plantillas ofrecidas:

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

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



4. Rotación de un vector hacia la derecha

Crear un subprograma que, dado un vector de N enteros (completo, o sea, con N enteros) cuyos valores pueden no estar ordenados, lo modifique rotando sus elementos una posición a la derecha, de modo que el último elemento pase a ser el primero, tal y como se muestra en el siguiente caso de prueba:

Vector de entrada:

1	2	3	4	5	6	7	8	9	10
67	45	23	78	12	40	55	24	89	34

Vector de salida:

1	2	3	4	5	6	7	8	9	10
34	67	45	23	78	12	40	55	24	89

Plantillas ofrecidas:

rotar_derecha.adb	Para hacer el subprograma en Ada
escribir_vector.adb	Para hacer un subprograma auxiliar en Ada
prueba_rotar_derecha.adb	Para completar y probar el subprograma Ada

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

5. Borrado del tercer elemento en lista no ordenada

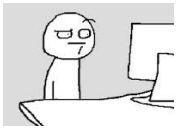
Crear un subprograma que, dado un vector de N enteros (no necesariamente completo, o sea, con, como mucho, N enteros terminado con el valor -1) cuyos valores pueden no estar ordenados, lo modifique eliminando su tercer elemento (si existe). Si la lista tiene menos de 3 elementos, entonces no se hará nada. Los elementos de la lista no tienen por qué estar ordenados (ni antes ni después del borrado), lo que se puede aprovechar para ganar en eficiencia al eliminar el tercer elemento.

NOTA: Es imprescindible que la solución sea eficiente. En concreto, se aprovechará que el orden de los elementos de la lista es indiferente (tanto antes como después de la eliminación del primer elemento) para borrar el tercer elemento sin utilizar bucles.

Plantillas ofrecidas:

eliminar_tercer_elemento.adb	Para hacer el subprograma en Ada
prueba_eliminar_tercer_elemento.adb	Para completar y probar el subprograma Ada

Fichero adicional: vectores.ads y escribir_vector.adb (No hay que modificarlo).



6. Borrado del tercer elemento en una lista ordenada

Crear un subprograma que, dado una lista de, como máximo N enteros cuyos valores están ordenados ascendentemente y terminado con el valor -1, lo modifique eliminando su tercer elemento (si existe, si no existe no hará nada). Este ejercicio es similar al anterior, con la diferencia de que en este caso los elementos de la lista están ordenados antes y deben seguir ordenados después de realizar el borrado del tercer elemento.

Plantillas ofrecidas:

eliminar_tercer_elemento_ordenada.adb	Para hacer el subprograma en Ada
prueba_eliminar_tercer_elemento_ordenada.adb	Para completar y probar el subprograma Ada

Fichero adicional: vectores.ads y escribir_vector.adb (No hay que modificarlo).

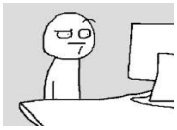
7. Inserción de un elemento en una posición

Crear un subprograma que, dados dos enteros *num* y *pos*, y una lista de, como máximo N-1 enteros (lo que significa que aún hay hueco para uno más) y terminado con el valor -1, inserte en la lista el número *num* en la posición *pos*, desplazando los elementos que corresponda una posición a la derecha.
Para resolver este problema conviene reutilizar el programa *rotar_derecha*, haciéndole una pequeña modificación y añadiéndole como parámetros de entrada la posición *Pos* y el número a insertar *Num*.

Plantillas ofrecidas:

insertar_elemento_en_pos.adb	Para hacer el subprograma en Ada
prueba_insertar_elemento_en_pos.adb	Para completar y probar el subprograma Ada

Fichero adicional: vectores.ads y escribir_vector.adb (No hay que modificarlo).



8. Búsqueda de la posición en una matriz

Hacer un subprograma que, dado un valor entero y una matriz de N x M enteros (N>0 y M>0), devuelva la posición en la que se encuentra ese valor entero (es decir, sus coordenadas Fila y Columna). Si el entero está repetido, se devolverá la posición de la primera aparición del entero en la matriz comenzando de arriba abajo y de izquierda a derecha, y si no está se devolverá como posición la fila -1 y la columna -1.

Plantillas ofrecidas:

posicion_en_matriz.adb	Para hacer el subprograma en Ada
prueba_posicion_en_matriz.adb	Para completar y probar el subprograma Ada

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

9. Máximo de una matriz

Crear un subprograma que, dada una matriz de enteros de dimensiones N x M (N,M>0), calcule el valor máximo y la posición en la que se encuentra. Si el valor máximo está repetido, devolver la posición de la primera de sus apariciones en la matriz contando por filas de arriba abajo y en cada fila de izquierda a derecha.

Plantillas ofrecidas:

calcular_maximo.adb	Para hacer el subprograma en Ada
prueba_calcular_maximo.adb	Para completar y probar el subprograma Ada

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