EJERCICIOS DE EXÁMENES

1. Implementar una función cuyos argumentos de entrada sean los valores de tres ángulos (medidos en grados) y que devuelva:

0 si los tres ángulos no pueden pertenecer a un triángulo

1 si los tres ángulos pueden pertenecer a un triángulo rectángulo

2 si los tres ángulos pueden pertenecer a un triángulo obtusángulo

3 si los tres ángulos pueden pertenecer a un triángulo acutángulo

Téngase en cuenta:

- la suma de los ángulos de un triángulo es 180º
- un triángulo rectángulo tiene un ángulo de 90º
- un triángulo obtusángulo tiene un ángulo mayor de 90º
- un triángulo acutángulo tiene los tres ángulos menores de 90º

---

2. Implementar una subrutina que calcule la derivada de un polinomio de a lo sumo grado 3.

Ambos polinomios (el de entrada y su derivada) se representarán mediante vectores de enteros de cuatro componentes, por ejemplo:

La representación del polinomio 7 -2x -8x^2 +6x^3 sería:

7	-2	-8	6

y la de su derivada -2 -16x+18x^2 sería:

-2	-16	18	0

Por tanto, la subrutina debe de tener dos argumentos de tipo vector, uno de entrada con la información del polinomio a derivar y otro de salida donde se calculará la derivada del polinomio.

---

3. Implementar una función cuyos argumentos de entrada sean los coeficientes de una ecuación de segundo grado y que devuelva:

0 si la ecuación no tiene soluciones reales

1 si la ecuación tiene una solución real doble

2 si la ecuación tiene dos soluciones reales diferentes

3 si la ecuación no es de segundo grado

Téngase en cuenta:

- La ecuación $ax^2 + bx + c = 0$ es de segundo grado si $a \neq 0$
- Si $ax^2 + bx + c = 0$ es una ecuación de segundo grado su discriminante es: $\Delta = b^2 - 4ac$
- Una ecuación de segundo grado no tiene soluciones reales si su discriminante es negativo.
- Una ecuación de segundo grado tiene una solución real doble si su discriminante es nulo.
- Una ecuación de segundo grado tiene dos soluciones reales diferentes si su discriminante es positivo.

---

4. Implementar una subrutina que calcule la integral de un polinomio de a lo sumo grado 2.

Ambos polinomios (el de entrada y su integral) se representarán mediante vectores de reales de cuatro componentes, por ejemplo:

La representación del polinomio $-2 -16x+18x^2$ sería

-2	-16	18	0

y la de su integral $-2x -8x^2 +6x^3$ sería:

0	-2	-8	6

Por tanto, la subrutina debe de tener dos argumentos de tipo vector, uno de entrada con la información del polinomio a integrar y otro de salida donde se calculará la integral del polinomio.

---

5. La versión internacional de la codificación de los dígitos en Morse es la siguiente:

1	.----	6	-....
2	..---	7	--...
3	...--	8	---..
4	....-	9	----.
5	.....	0	-----

Implementar una función en Fortran que reciba como argumento de entrada la representación de un digito en código Morse y devuelva como resultado el dígito descodificado. El digito codificado viene representado mediante un vector de caracteres que contiene una sucesión de puntos y rayas terminada con un espacio en blanco. Por ejemplo:

.	.	.	-	-		→ 3

---

6. Los resultados de una liga deportiva a doble vuelta de N equipos se pueden representar mediante una matriz de caracteres. Cada resultado se corresponde con un elemento de la matriz cuya fila identifica al equipo local y la columna al visitante. Los caracteres utilizados y su significado son:

'1' → vence el equipo local

'X' → empate entre ambos equipos

'2' → vence el equipo visitante

Implementar una subrutina en lenguaje Fortran que reciba una matriz (tal como se ha descrito anteriormente) con los resultados de una liga de 20 equipos. La subrutina tendrá como argumento de salida un vector con las puntuaciones conseguidas por cada equipo teniendo en cuenta que las victorias aportan 3 puntos, los empates 1 y las derrotas 0.

El siguiente ejemplo muestra una matriz con los resultados de una liga de 4 equipos y el vector con las puntuaciones obtenidas por cada uno de ellos.

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} & 2 & X & 1 \\ X & & 1 & 2 \\ 2 & 2 & & 1 \\ X & 1 & 1 & \end{pmatrix} \end{array} \Rightarrow \begin{pmatrix} 9 \\ 10 \\ 4 \\ 10 \end{pmatrix}$$

---

7. Implementar una función en fortran que dada una matriz de 3x3 que contiene ***X***, ***O*** y – determine si se ha conseguido tres en raya y quien lo ha conseguido. Por ejemplo:

$$\begin{matrix} O & X & - \\ X & O & O \\ X & - & O \end{matrix}$$

En este caso el ganador será el jugador de la ***O***.

La función debe recibir como parámetro la matriz de caracteres y devolver un carácter:

- **E** si ninguno ha conseguido tres en raya,
- **X** si el jugador de la **X** ha conseguido tres en raya,
- **O** si el jugador de la **O** ha conseguido tres en raya.

Tener en cuenta que:

- se puede conseguir tres en raya en horizontal (1ª , 2ª o 3ª línea), en vertical (1ª , 2ª o 3ª columna) o en las dos diagonales,
- solo uno de los jugadores puede conseguir tres en raya.

Se valorará positivamente la utilización de constantes para generalizar la función para matrices de *n x n*, es decir, la posibilidad de jugar al *n* en raya.

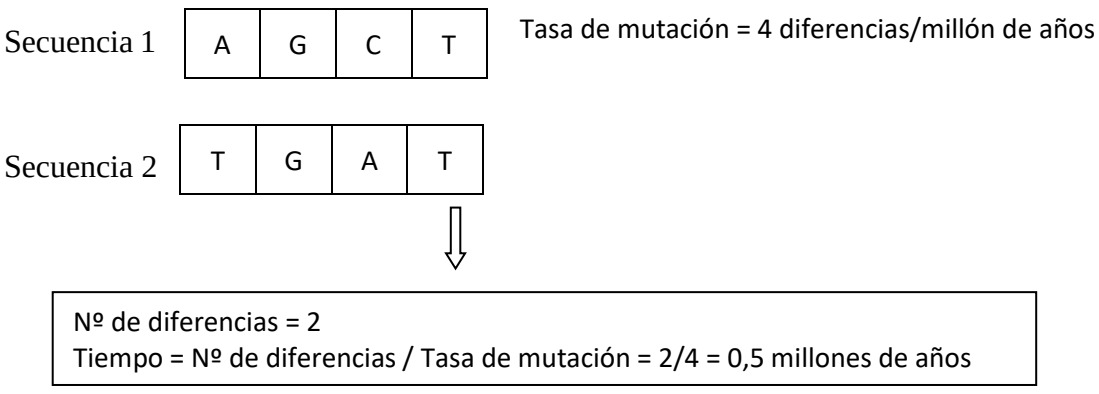
---

8. Con el fin de determinar el tiempo en que divergieron dos especies se comparan sus secuencias de ADN.

El número de diferencias encontradas, comparando posición a posición, dividido entre la tasa de mutación (nº diferencias /millones de años) proporciona la cantidad de tiempo en que ambas especies se separaron.

En una versión simplificada de este problema se pide implementar una subrutina en lenguaje Fortran que reciba dos secuencias de ADN (representadas por dos vectores de caracteres de tamaño 100) y la tasa de mutación. Como resultado proporcionará en dos argumentos de salida el número de diferencias entre ambas secuencias y el tiempo transcurrido desde que las dos especies divergieron.

Ejemplo con dos secuencias de tamaño 4:



El tres en raya es un juego entre dos jugadores (O y X) que marcan los espacios de un tablero de 3×3 alternadamente. Un jugador gana si consigue tener una línea de tres de sus símbolos. La línea puede ser horizontal, vertical o diagonal.

9. Codificar una función en Fortran que a partir de un tablero de 3 en raya, el jugador que tiene el turno y una línea concreta (columna, fila o diagonal), devuelva un valor negativo (por ejemplo -1) si no es posible alcanzar situación de victoria por ese jugador en esa línea. En caso contrario, la función devolverá el nº de fichas que el jugador tiene colocadas en dicha línea.

Consideraciones:

En una línea no se puede alcanzar 3 en raya por un jugador si en la misma aparecen fichas de su adversario.

Los jugadores quedan identificados por el tipo de ficha que utilizan ('O' , 'X').

La línea se identifica con un valor entero del 1 al 8. Del 1 al 3 para las filas 1ª, 2ª y 3ª respectivamente, del 4 al 6 para las columnas 1ª, 2ª y 3ª respectivamente, el 7 para la diagonal principal y el 8 para la secundaria.

Las fichas en el tablero se representan con los caracteres 'O' y 'X' y las posiciones libres por el espacio en blanco.

El código debe ser genérico en el sentido de poder desarrollarse en un tablero de n x n ("n en raya"). Con tan sólo modificar el valor de una constante (en el caso del "3 en raya" el valor de la constante es 3).

Ejemplo 1:

O	X	
O		X
		X

, 'O' , 1 → -1

Interpretación: En la línea 1 (1ªfila) no se puede conseguir 3 en raya con fichas 'O'

Tablero, jugador, línea
(argumentos de entrada)

Ejemplo 2:

O	X	
O		X
		X

, 'X' , 6 → 2

Interpretación: En la línea 6 (3ª columna) se puede conseguir 3 en raya con fichas 'X' y de momento hay colocadas 2

Tablero, jugador, línea
(argumentos de entrada)

De forma opcional los alumnos que deseen mejorar la calificación obtenida en la prueba de la traza de la evaluación continua pueden realizar la traza de su implementación con los dos ejemplos.

10. Implementar una subrutina en Fortran que, a partir de un tablero de tres en raya recibido como argumento y apoyándose en la función del ejercicio anterior, devuelva, en caso de haberse alcanzado situación de victoria, la línea donde se ha conseguido y el jugador que ha vencido. En caso de que no exista situación de 3 en raya se devolverá un valor negativo en el argumento que indica la línea y el espacio en blanco en el argumento que indica el jugador.

Ejemplo 1:

O	X	
O		X
O		X

→ 4 , 'O'

Interpretación: Hay situación de 3 en raya en la línea 4 (1ª columna) y la ha conseguido el jugador que juega con el tipo de ficha 'O'

Tablero (argumento de entrada) — línea, jugador (argumentos de salida)

Ejemplo 2:

O	X	
O		X
		X

→ -1 , ' '

Interpretación: No hay situación de 3 en raya

Tablero (argumento de entrada) — línea, jugador (argumentos de salida)

Además de ser válidas las consideraciones del ejercicio anterior ha de tenerse en cuenta que se puede llamar a la función del ejercicio 1., sin necesidad de haberla codificado, una vez se haya definido su cabecera.

11. Implementar una función cuyos argumentos de entrada sean dos polinomios y que devuelva:

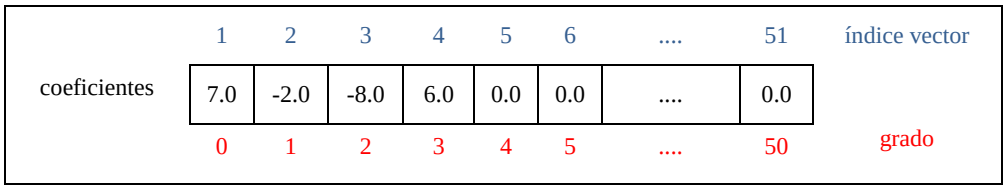
- -1 si el primer polinomio tiene mayor grado que el segundo
- -2 si el segundo polinomio tiene mayor grado que el primero
- El grado de los polinomios en caso de que sea el mismo en ambos.

Ejemplos:

- Dados los polinomios $-2x -8x^2 +6x^3$ y $-2 -16x+18x^2$ la función devolverá -1
- Dados los polinomios $-2 -16x+18x^2$ y $-2x -8x^2 +6x^3$ la función devolverá -2
- Dados los polinomios $-2x -8x^2 +6x^3$ y $-2 -16x+18x^2-9x^3$ la función devolverá 3

La forma de representar los polinomios se corresponde con el siguiente ejemplo:

La representación del polinomio ***7 - 2x - 8x² + 6x³*** sería:



La componente 1 del vector contendrá el coeficiente correspondiente al monomio de grado 0 del polinomio, la componente 2 el coeficiente de grado 1,..., hasta completar el vector. Si no existe monomio correspondiente a un grado su coeficiente será 0.

El grado máximo representable debe ser 50 y se reflejará en una constante.

---

12. Implementar una subrutina que calcule el límite de un cociente de polinomios cuando x -> ∞.

$$\lim_{x\to\infty} \frac{P(x)}{Q(x)}$$

Recibirá dos polinomios como argumentos de entrada y los argumentos de salida serán tres: el signo del límite, el valor del límite para los casos en que este acotado (es decir, no sea ∞) y otro que tomará el valor '∞' en los casos en que el límite no esté acotado, y ' ' en el caso contrario.

Ejemplos:

- Dados los polinomios $-2x -8x^2 +6x^3$ y $-2 -16x+18x^2$ la subrutina devolverá: +, , '∞'
- Dados los polinomios $-2 -16x+18x^2$ y $-2x -8x^2 +6x^3$ la subrutina devolverá: +, 0, ' '
- Dados los polinomios $-2x -8x^2 +6x^3$ y $-2 -16x+18x^2-9x^3$ la subrutina devolverá: -, 0,66..,' '

Téngase en cuenta:

1. Se debe llamar a la función del primer ejercicio para la resolución de este
2. El signo del límite de un cociente de polinomios cuando x -> ∞ será positivo si coinciden los signos de los respectivos monomios de mayor grado de ambos polinomios y negativo en caso contrario.
3. El valor será ∞ si el grado del polinomio numerador es mayor que el del polinomio denominador.
4. El valor será 0 si el grado del polinomio numerador es menor que el del polinomio denominador.
5. El valor será el cociente de los respectivos coeficientes de mayor grado de los polinomios, prescindiendo del signo, si coinciden los grados de los mismos.

Se dice que un **número de N dígitos** es *"narcisista"* si resulta ser igual a la **suma de las potencias de orden N de sus dígitos**. Por ejemplo, los números 153, 1634 y 54748 son *"narcisistas"*:

$$153 = 1^3+5^3+3^3$$
$$1634 = 1^4+6^4+3^4+4^4$$
$$54748 = 5^5+4^5+7^5+4^5+8^5$$

13. Codificar una función en Fortran que indique si un número es "narcisista". El número vendrá representado por dos argumentos: un vector cuyas componentes se corresponderán con los dígitos del número y otro argumento que contendrá la cantidad de dígitos significativos del número.

Ejemplos:
Los argumentos correspondientes al número 153 son:

10	9	8	7	6	5	4	3	2	1	
0	0	0	0	0	0	0	1	5	3	3

Los argumentos correspondientes al número 1634 son:

10	9	8	7	6	5	4	3	2	1	
0	0	0	0	0	0	1	6	3	4	4

Los argumentos correspondientes al número 54748 son:

10	9	8	7	6	5	4	3	2	1	
0	0	0	0	0	5	4	7	4	8	5

El código debe ser genérico en el sentido de poder desarrollarse en un vector de N componentes con tan sólo modificar el valor de una constante (en el ejemplo el valor de la constante es 10).
De forma opcional los alumnos que deseen mejorar la calificación obtenida en la prueba de la traza de la evaluación continua pueden realizar la traza de su implementación con los números 370 y 371.

---

14. Implementar una subrutina en Fortran que reciba como argumentos dos números y que apoyándose en la función del ejercicio 1., imprima por pantalla los números "narcisistas" que hay entre ellos.

Además de ser válidas las consideraciones del ejercicio anterior ha de tenerse en cuenta que se puede **llamar a la función** del ejercicio 1., sin necesidad de haberla codificado, **una vez se haya definido su cabecera**.

Nota: En ambos ejercicios se pueden implementar los subprogramas que se consideren necesarios.

15. Dado un número N, su crápulo es un número que se obtiene de la siguiente forma: se suman los dígitos de N, si el valor resultante tiene un único dígito ese es el crápulo y si tiene más entonces se vuelven a sumar los dígitos del resultado hasta que la suma sea un único dígito. Por ejemplo:

- 5 → crápulo 5
- 52 → crápulo 7
  5+2=7
- 97 → crápulo 7
  9+7=16 → 1+6=7
- 9859 → crápulo 1
  9+8+5+6=28 → 2+8=10 → 1+0=1

Implementar una función en frotran que reciba como argumento un número entero y que devuelva su crápulo (téngase en cuenta que se recibe el número completo y no sus dígitos)

---

16. Las cadenas de ADN están formadas por una disposición secuencial de las bases:

A → adenina, T → timina, C → citosina o G → guanina

Por ejemplo, una secuencia de ADN puede ser ATGCTAGATCGC...

Dos cadenas de ADN serán más similares cuantas más bases coincidan en ambas posición a posición. También se tiene en cuenta las bases coincidentes en ambas cadenas que no están en las mismas posiciones.

En una versión simplificada de este problema se pide implementar una subrutina en lenguaje Fortran que reciba dos secuencias de ADN (de tamaño 100). Como resultado proporcionará en dos argumentos de salida el número de coincidencias entre ambas secuencias en las mismas posiciones y el número de coincidencias en diferentes posiciones.

Ejemplo con dos secuencias de tamaño 4:

Secuencia 1	A	G	T	A
Secuencia 2	T	G	A	T

⇩

Nº de coincidencias misma posición = 1
Nº de coincidencias distinta posición = 2

El código debe ser genérico en el sentido de poder desarrollarse con diferentes longitudes de las secuencias de ADN con tan sólo modificar el valor de una constante.

Nota: En ambos ejercicios se pueden implementar los subprogramas que se consideren necesarios

Dos números ***amigos*** son dos enteros positivos (n1, n2) tales que la suma de los divisores propios de uno de ellos es igual al otro y viceversa (la unidad se considera divisor propio, pero no lo es el mismo número).

Por ejemplo, 220 y 284 son amigos, ya que:

- Suma de divisores de 284: 1 + 2 + 4 + 71 + 142 = 220
- Suma de divisores de 220: 1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284

También son números amigos 1.184 y 1.210

17. Implementar una función en lenguaje Fortran que reciba como **argumentos de entrada dos números enteros** y que **devuelva si son o no números amigos**.

---

18. Implementar una subrutina en lenguaje Fortran que, **utilizando la función del ejercicio anterior**, calcule los **números amigos** que hay en un **intervalo dado**. La subrutina debe recibir como **argumentos de entrada dos números enteros** que representan el intervalo y devolverá los números amigos en una **matriz de 2 columnas y n filas (argumento de salida)**, de forma que cada fila guardará dos números amigos contenidos en el intervalo. Ejemplo:

Si consideramos el intervalo (100,2.000) la subrutina deberá devolver la siguiente matriz

	1	2
1	220	284
2	1.184	1.210
3	0	0
.	.	.
.	.	.

Ha de tenerse en cuenta que se puede **llamar a la función** del ejercicio 1, sin necesidad de haberla codificado, **una vez se haya definido su cabecera**.

Nota: En ambos ejercicios se pueden implementar los subprogramas que se consideren necesarios.

Un *centro numérico* es un número que separa una secuencia de números enteros consecutivos (comenzando en 1) en dos sub-secuencias de números, cuyas sumas son iguales. El primer *centro numérico* es el 6, el cual separa la secuencia (1 a 8) en las secuencias: (1, 2, 3, 4, 5) y (7, 8) cuyas sumas son ambas iguales a 15. El segundo *centro numérico* es el 35, el cual separa la secuencia (1 a 49) en las secuencias: (1 a 34) y (36 a 49) cuyas sumas son ambas iguales a 595.

19. Escribir una **subrutina** en fortran que reciba como **argumento de entrada un número entero** y devuelva **como argumento de salida un vector** que contenga los *centros numéricos* entre 1 y ése número dado.
Ejemplo:

num=1000 → **centrosNum**

6	35	204	0	0	...	0

---

20. Escribir una **función** en fortran que reciba como **argumento de entrada un número entero** y que, una vez encontrado el mayor *centro numérico* menor o igual que éste argumento de entrada, **devuelva el extremo superior** de la secuencia de la que es centro numérico. Para encontrar este centro numérico **deberá utilizarse la subrutina del ejercicio 1**, sin necesidad de haberla codificado, una vez se haya definido su cabecera.
Ejemplos:

- Para **n=10** el **centro numérico mayor** es el **6** por lo que tendrá que **devolver 8** extremo superior de la secuencia **(1 a 8)**. Ver ejemplos en definición de centro numérico.
- Para **n=100** el **centro numérico mayor** es el **35** por lo que tendrá que **devolver 49** extremo superior de la secuencia **(1 a 49)**. Ver ejemplos en definición de centro numérico.

Nota: En ambos ejercicios se pueden implementar los subprogramas que se consideren necesarios

En diversas variantes del **juego de las damas** se da situación de captura cuando, en turno de juego, un peón encuentra en casilla diagonal contigua, en el sentido de avance, a otra ficha del contrario estando la posterior vacía. En ese caso puede saltar por encima del peón oponente hasta la casilla vacía, retirándolo del tablero.

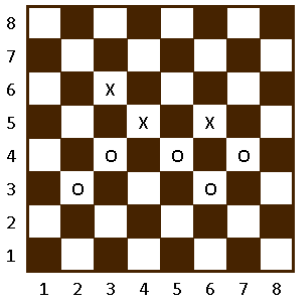
21. Implementar una **función** en lenguaje Fortran que reciba **como argumentos de entrada:**

- la posición de un peón (fila, columna), que se supone en turno de juego,
- el tipo del peón ('O' si es blanco, 'X' si es negro)
- y el tablero con la situación de la partida,

y que **devuelva** si dicho peón está en situación de capturar alguna ficha oponente.

Se entenderá que el sentido de avance se corresponde con el de la numeración creciente de las filas. Por ejemplo, si un peón está en la fila 4 sólo podría capturar a fichas oponentes de la fila 5 de cumplirse los requisitos indicados.

Ejemplo:



Para turno de juego blancas, 'O':

- Peón (3,2) ***no está*** en situación de capturar

- Peón (4,3) ***está*** en situación de capturar

---

22. Implementar una subrutina en lenguaje Fortran que reciba como argumentos de entrada:

- un tablero con la situación de una partida
- y el tipo de ficha que tiene el turno de juego ('O' o 'X')

y que **devuelva**, **utilizando la función del ejercicio anterior**:

- el número de fichas en situación de capturar
- y las situaciones de capturas, es decir, que indique las posiciones de los peones que pueden comer a fichas adversarias (las coordenadas deberán devolverse en **dos vectores**, uno para las posiciones de las filas y otro para el de columnas).

Ha de tenerse en cuenta que se puede **llamar a la función** del ejercicio 1, sin necesidad de haberla codificado, **una vez se haya definido su cabecera**.

Ejemplo:

Siguiendo el ejemplo del ejercicio 1, las blancas, 'O', ***tienen 3 situaciones de captura*** en las posiciones:

	1	2	3	.	.	.	12
VectorFila	4	4	4	.	.	.	.

	1	2	3	.	.	.	12
VectorColumna	3	5	7	.	.	.	.

Notas:

- El código debe ser genérico en el sentido de poder desarrollarse en un tablero de n x n con tan sólo modificar el valor de una constante.
- En ambos ejercicios se pueden implementar los subprogramas que se consideren necesarios.