**EXAMEN FINAL FUNDAMENTOS PROGRAMACIÓN**
**JUNIO 2023/2024**

***Normas:***

- *Duración prevista del examen: 2h 00 min.*
- *El fichero entregado debe ser un fichero válido de Python (NO una sesión interactiva de Jupyter), es decir, debe tener la extensión .py.*
- *Debéis subir todas las funciones de los ejercicios en un solo fichero.*
- *Las funciones implementadas deben respetar la signatura definida en el ejercicio (nombre, parámetros y valores de retorno correctos).*
- *La entrega se encuentra abierta en https://pracdlsi.dlsi.ua.es/. Está prohibido acceder a cualquier otra URL durante el examen.*
- *Recordad que la red puede ir lenta, y que el servidor se cierra exactamente a la hora de finalización del examen: reservad al menos 5 minutos al final del examen para la entrega.*
- *No olvidéis indicar al inicio del fichero vuestro **dni** y nombre mediante un comentario Python*
- *En el cuerpo de las funciones **no debe haber ninguna sentencia de tipo input ni print.***
    - *Si queréis que el ejercicio incluya vuestro código de prueba, lo debéis poner al final del archivo .py, dentro del bloque, incluido en la plantilla del examen:*

```
if __name__ == "__main__":
    pass
    #aquí vuestro código de prueba
```

    - *Notad como tanto name como main van precedidos y sucedidos de DOS guiones bajos.*

- *No entreguéis código que contenga errores sintácticos: abortará el corrector y eso supondrá un 0 en el ejercicio.*
- *Podéis ir **guardando** vuestro código en la **unidad D: o E:** (cambia según el ordenador usado). De este modo, si se os reiniciase por cualquier motivo, no se perderá vuestro trabajo. **Borra todo** el contenido de esa unidad al acabar el examen.*
- *Si se detecta que el código entregado ha sido **plagiado** se procederá a actuar según el artículo 14 del Reglamento para la Evaluación de los Aprendizajes de la Universidad de Alicante (BOUA 9/12/2015)*

***Ejercicio 1 (5 puntos):***

Una clínica especializada en estudios genéticos ha recogido información genética de pacientes en un archivo llamado **`datos_geneticos.txt`**. Este archivo contiene datos de mutaciones, donde cada línea describe un paciente y sus mutaciones conocidas en el formato **`ID_Paciente:Mutación1,Mutación2,...,MutaciónN`**.

Ejemplo de archivo `datos_geneticos.txt`:

```
12345:BRCA1,MDM2,P53
67890:BRCA1,CDK2
54321:P53,CDK2,MDM2
45234:
98765:BRCA1,P53
```

La clínica necesita realizar un análisis para determinar cuáles son las mutaciones más comunes entre los pacientes y desea que los resultados sean almacenados en un nuevo archivo llamado `reporte_mutaciones.txt`, que contendrá cada mutación y la cantidad de veces que aparece, ordenadas de mayor a menor frecuencia.

Se pide:

**a)** (2.5 puntos) Crea una función `get_frec_mutaciones(fich_entrada)` que reciba el nombre del fichero de entrada y devuelva un diccionario con la frecuencia de cada mutación, donde la clave será el nombre de la mutación, y el valor el contador correspondiente. **El procesamiento del fichero debe ignorar cualquier línea vacía del fichero. También debe ignorar cualquier espacio en blanco o tabulador que pueda estar rodeando los datos.**

E.g. para el fichero datos_geneticos.txt, la función debería devolver el siguiente diccionario: `{'BRCA1': 3, 'MDM2': 2, 'P53': 3, 'CDK2': 2}`

Si el fichero está vacío, la función devolverá un diccionario vacío. Si el fichero no existe, la función devolverá `-1`, y ante cualquier otra excepción, la función devolverá `-2`.

Recordad además que puede haber pacientes que no presenten ninguna mutación genética, y esto no es ningún error del fichero, por lo que debe procesarse correctamente.

b) (2.5 puntos): Crea otra función `guarda_frec_mutaciones(dic,fich_salida)` que reciba el diccionario con los contadores de las distintas mutaciones y guarde en fich_salida el contenido de dicho diccionario, con las mutaciones ordenadas de mayor a menor frecuencia. En caso de que dos mutaciones tengan la misma frecuencia, el fichero mostrará en la misma línea todas las mutaciones de esa frecuencia separadas por comas.

E.g. para el diccionario anterior, el fichero generado sería:

```
BRCA1,P53:3
MDM2,CDK2:2
```

*Nota: Para obtener el contenido del diccionario ordenado, podéis transformarlo a lista y ordenar la lista en su lugar.*

### *Ejercicio 2*

En un laboratorio de biotecnología, es esencial calcular la masa molecular de polímeros biológicos, especialmente aquellos utilizados en la creación de nuevos materiales biocompatibles. Algunos de estos polímeros se componen de múltiples unidades repetidas de un solo monómero específico. La masa de un polímero se calculará como la masa de su monómero multiplicado por el número de monómeros que componen el polímero.

Se pide:

- (3 puntos) Escribe una función `parsea_formula_monomero(formula)` que, dada la fórmula de un monómero en formato cadena, devuelva un diccionario donde las claves serán los elementos químicos que componen el monómero y los valores serán el número de átomos de ese elemento que tiene el monómero. La cadena del monómero será del estilo "C2-H4" (etileno). Si un monómero solo tiene un átomo de un elemento, el número puede estar omitido, e.g. H-C-N (cianuro de hidrógeno)
    - **La función lanzará un error de tipo 'ElementoRepetidoError' (que no capturará) si** tipo de átomo ha sido especificado más de una vez. Por ejemplo, **H3-H2**
    - **La función lanzará un error de tipo 'ElementoNoExisteError' (que no capturará)** si un tipo de átomo no aparece en el diccionario de masas. Por ejemplo, **Tz5-H3**
    - **La función lanzará un error de tipo 'FormMonomeroError (que no capturará)** si el monómero incluye algún carácter incorrecto (algo que no sean letras, números y el separador "-"): Por ejemplo, **H2.O** o **N?H3**

Las tres excepciones se encuentran definidas en la plantilla del examen.

Ejemplo: `parsea_formula_monomero("C2-H4")` devolverá el diccionario {'C': 2, 'H': 4}.

Ejemplo: `parsea_formula_monomero("C-H4")` devolverá el diccionario {'C': 1, 'H': 4}

- (2 puntos) Escribe una función llamada `calcula_masa_polimero (dmon,dmasas,num_mon)` que reciba tres parámetros: un diccionario que indica elementos y número de átomos de ese elemento que contiene el monómero (e.g. el que habéis obtenido mediante la función anterior), otro diccionario que indica la masa de cada elemento (diccionario `dmasas`, proporcionado en la plantilla) y un entero que indique el número de unidades repetidas de ese monómero en el polímero. La función debe calcular la masa molecular del **polímero** y devolverlo como un **float redondeado a dos decimales**. Si el diccionario está vacío, la función devolverá 0. Si el número de polímeros NO es un entero mayor que 0, la función devolverá -1. Recordad: la masa molecular de un polímero se calcula como la masa molecular del monómero multiplicada por el número de monómeros que conforman el polímero.

Ejemplo: Para calcular la masa del polietileno de baja densidad, asumiendo que cada molécula de polietileno contiene 4000 monómeros de etileno, debemos llamar a la función `calcula_masa_polimero ({'C': 2, 'H': 4},dmasas,4000)`. Esta función devolverá 112206.40