Ejercicios de Control de Flujo (Condicionales y Bucles)

1. **Número par o impar:**
   Escribe un programa que pida un número al usuario y determine si es par o impar.
2. **Mayor de tres números:**
   Escribe un programa que pida tres números y determine cuál es el mayor de ellos.
3. **Contar números positivos y negativos:**
   Escribe un programa que reciba una lista de números y cuente cuántos son positivos, negativos y cuántos son cero.
4. **Divisible por 3 y 5:**
   Escribe un programa que determine si un número es divisible por 3 y 5.
5. **Calificación de un estudiante:**
   Escribe un programa que reciba una calificación (0-100) y determine si la calificación es A, B, C, D o F según los rangos establecidos.
6. **Multiplicación con bucles:**
   Escribe un programa que muestre la tabla de multiplicar de un número utilizando un bucle.
7. **Números primos:**
   Escribe un programa que determine si un número es primo o no.
8. **Imprimir números impares del 1 al 100:**
   Escribe un programa que imprima todos los números impares entre 1 y 100.
9. **Suma de números hasta que el usuario ingrese un número negativo:**
   Escribe un programa que permita al usuario ingresar números y calcule la suma hasta que se ingrese un número negativo.
10. **Adivina el número:**
    Escribe un programa que genere un número aleatorio entre 1 y 100, y pida al usuario adivinarlo, indicando si el número es mayor o menor que el número secreto.

Ejercicios de Manipulación de Cadenas

11. **Contar vocales en una cadena:**
    Escribe un programa que reciba una cadena de texto y cuente cuántas vocales contiene.
12. **Revertir una cadena:**
    Escribe un programa que reciba una cadena y la imprima en orden inverso.
13. **Palíndromo:**
    Escribe un programa que determine si una cadena es un palíndromo (que se lee igual de izquierda a derecha que de derecha a izquierda).
14. **Contar palabras en una cadena:**
    Escribe un programa que cuente cuántas palabras contiene una cadena de texto.
15. **Reemplazo de subcadenas:**
    Escribe un programa que reciba una cadena y reemplace todas las ocurrencias de una palabra por otra.
16. **Eliminar espacios en blanco al principio y al final de una cadena:**
    Escribe un programa que reciba una cadena y elimine los espacios en blanco al principio y al final.

17. **Convertir una cadena a mayúsculas y minúsculas:**
    Escribe un programa que reciba una cadena y la convierta completamente a mayúsculas y luego a minúsculas.
18. **Contar caracteres únicos en una cadena:**
    Escribe un programa que cuente cuántos caracteres distintos aparecen en una cadena.
19. **Contar el número de ocurrencias de una palabra en una cadena:**
    Escribe un programa que cuente cuántas veces aparece una palabra específica en una cadena.
20. **Concatenación de cadenas:**
    Escribe un programa que reciba dos cadenas y las una en una sola.

Ejercicios de Arreglos/Listas

21. **Suma de elementos en una lista:**
    Escribe un programa que reciba una lista de números y calcule su suma.
22. **Promedio de una lista de números:**
    Escribe un programa que reciba una lista de números y calcule su promedio.
23. **Buscar un número en una lista:**
    Escribe un programa que reciba una lista de números y determine si un número específico se encuentra en la lista.
24. **Ordenar una lista de números de menor a mayor:**
    Escribe un programa que ordene una lista de números de menor a mayor.
25. **Eliminar elementos duplicados de una lista:**
    Escribe un programa que reciba una lista y elimine los elementos duplicados.
26. **Intercambiar dos elementos de una lista:**
    Escribe un programa que reciba una lista de números y los intercambie entre dos posiciones específicas.
27. **Encontrar el valor máximo y mínimo en una lista:**
    Escribe un programa que reciba una lista de números y determine cuál es el valor máximo y mínimo.
28. **Invertir una lista:**
    Escribe un programa que reciba una lista de números y la imprima en orden inverso.
29. **Contar ocurrencias de un número en una lista:**
    Escribe un programa que reciba una lista de números y cuente cuántas veces aparece un número específico.
30. **Multiplicar todos los elementos de una lista:**
    Escribe un programa que reciba una lista de números y calcule el producto de todos los elementos.

Ejercicios de Funciones

31. **Función para calcular el factorial:**
    Escribe una función que reciba un número y calcule su factorial.
32. **Función para verificar si un número es primo:**
    Escribe una función que reciba un número y devuelva `True` si es primo, o `False` si no lo es.

33. **Función para calcular la suma de los primeros n números:**
    Escribe una función que reciba un número n y calcule la suma de los primeros n números enteros.
34. **Función para calcular el área de un círculo:**
    Escribe una función que reciba el radio de un círculo y calcule su área.
35. **Función que reciba una lista y retorne su promedio:**
    Escribe una función que reciba una lista de números y retorne su promedio.
36. **Función que reciba dos números y retorne el mayor:**
    Escribe una función que reciba dos números y retorne el mayor de los dos.
37. **Función que reciba un número y retorne si es par o impar:**
    Escribe una función que reciba un número y retorne una cadena que indique si es "Par" o "Impar".
38. **Función para calcular la potencia de un número:**
    Escribe una función que reciba una base y un exponente, y calcule la potencia (base^exponente).
39. **Función que reciba una cadena y cuente las palabras:**
    Escribe una función que reciba una cadena de texto y cuente cuántas palabras contiene.
40. **Función para invertir una lista:**
    Escribe una función que reciba una lista y la invierta.

Ejercicios sobre Estructuras de Datos (Diccionarios, Tuplas, Conjuntos)

41. **Uso de diccionario para contar palabras:**
    Escribe un programa que reciba una cadena de texto y cuente cuántas veces aparece cada palabra, usando un diccionario.
42. **Eliminar una clave de un diccionario:**
    Escribe un programa que reciba un diccionario y elimine una clave específica proporcionada por el usuario.
43. **Buscar una clave en un diccionario:**
    Escribe un programa que reciba un diccionario y busque si una clave específica existe, imprimiendo un mensaje adecuado.
44. **Tupla con el máximo y mínimo:**
    Escribe un programa que reciba una lista de números y devuelva una tupla con el valor máximo y mínimo de la lista.
45. **Concatenación de tuplas:**
    Escribe un programa que reciba dos tuplas y las combine en una sola.
46. **Eliminar un elemento de un conjunto:**
    Escribe un programa que reciba un conjunto y elimine un elemento proporcionado por el usuario.
47. **Intersección de dos conjuntos:**
    Escribe un programa que reciba dos conjuntos y calcule su intersección (elementos comunes).
48. **Unión de dos conjuntos:**
    Escribe un programa que reciba dos conjuntos y calcule su unión (todos los elementos de ambos conjuntos, sin duplicados).

49. **Contar elementos únicos de una lista usando un conjunto:**
    Escribe un programa que reciba una lista de elementos y cuente cuántos elementos distintos contiene, usando un conjunto.
50. **Crear un diccionario de números al cuadrado:**
    Escribe un programa que genere un diccionario con claves como los números del 1 al 10 y sus valores como el cuadrado de esos números.

Ejercicios sobre Manejo de Archivos

51. **Leer un archivo de texto:**
    Escribe un programa que lea el contenido de un archivo de texto y lo imprima en pantalla.
52. **Escribir en un archivo de texto:**
    Escribe un programa que reciba una cadena de texto y la escriba en un archivo de texto llamado `salida.txt`.
53. **Contar líneas en un archivo:**
    Escribe un programa que lea un archivo de texto y cuente cuántas líneas tiene.
54. **Buscar una palabra en un archivo:**
    Escribe un programa que busque una palabra específica dentro de un archivo y diga cuántas veces aparece.
55. **Leer y escribir en un archivo CSV:**
    Escribe un programa que lea los datos de un archivo CSV y los imprima en pantalla.
56. **Eliminar líneas vacías de un archivo:**
    Escribe un programa que lea un archivo de texto y elimine todas las líneas vacías, guardando el resultado en otro archivo.
57. **Modificar una línea en un archivo:**
    Escribe un programa que modifique una línea específica en un archivo de texto.
58. **Copiar contenido de un archivo a otro:**
    Escribe un programa que copie el contenido de un archivo a otro archivo.
59. **Contar palabras en un archivo:**
    Escribe un programa que cuente cuántas veces aparece una palabra específica en un archivo de texto.
60. **Leer y procesar un archivo de log:**
    Escribe un programa que lea un archivo de log y muestre el número de errores registrados (cada línea contiene la palabra "ERROR").

Ejercicios sobre Algoritmos de Búsqueda y Ordenación

61. **Ordenar una lista con el algoritmo de burbuja:**
    Escribe un programa que ordene una lista de números usando el algoritmo de burbuja.
62. **Ordenar una lista con el algoritmo de selección:**
    Escribe un programa que ordene una lista de números usando el algoritmo de selección.
63. **Ordenar una lista con el algoritmo de inserción:**
    Escribe un programa que ordene una lista de números usando el algoritmo de inserción.

64. **Búsqueda lineal en una lista:**
    Escribe un programa que realice una búsqueda lineal en una lista para encontrar un número.
65. **Búsqueda binaria en una lista ordenada:**
    Escribe un programa que realice una búsqueda binaria en una lista ya ordenada para encontrar un número.
66. **Ordenar una lista de cadenas por longitud:**
    Escribe un programa que ordene una lista de cadenas de texto según su longitud, de menor a mayor.
67. **Búsqueda de un valor en un diccionario:**
    Escribe un programa que busque un valor en un diccionario dado un valor de clave.
68. **Encontrar el segundo número más grande en una lista:**
    Escribe un programa que encuentre el segundo número más grande en una lista de números.
69. **Mezclar dos listas ordenadas:**
    Escribe un programa que reciba dos listas ordenadas y las mezcle en una sola lista también ordenada.
70. **Eliminación de elementos de una lista si son menores que un valor:**
    Escribe un programa que elimine todos los elementos de una lista que sean menores que un valor dado.

Ejercicios sobre Recursión

71. **Calcular el factorial de un número (recursión):**
    Escribe una función recursiva que calcule el factorial de un número.
72. **Serie Fibonacci (recursión):**
    Escribe una función recursiva que devuelva el n-ésimo número de la serie Fibonacci.
73. **Contar dígitos de un número (recursión):**
    Escribe una función recursiva que reciba un número y cuente cuántos dígitos tiene.
74. **Potencia de un número (recursión):**
    Escribe una función recursiva que calcule la potencia de un número (base^exponente).
75. **Sumar los elementos de una lista (recursión):**
    Escribe una función recursiva que reciba una lista de números y calcule la suma de todos los elementos.
76. **Inverso de una cadena (recursión):**
    Escribe una función recursiva que reciba una cadena y la invierta.
77. **Imprimir una lista en orden inverso (recursión):**
    Escribe una función recursiva que reciba una lista y la imprima en orden inverso.
78. **Buscar un elemento en una lista (recursión):**
    Escribe una función recursiva que busque un elemento en una lista y devuelva si existe o no.
79. **Calcular el máximo de una lista (recursión):**
    Escribe una función recursiva que reciba una lista y calcule su valor máximo.
80. **Eliminar un elemento de una lista (recursión):**
    Escribe una función recursiva que reciba una lista y elimine un elemento específico.

Ejercicios sobre Programación Orientada a Objetos (POO)

81. **Crear una clase "Persona":**
    Escribe una clase `Persona` con atributos como nombre, edad y género. Luego, crea un objeto de esta clase y muestra sus atributos.
82. **Método en una clase para cambiar el nombre:**
    Escribe una clase `Persona` con un método para cambiar el nombre de la persona.
83. **Herencia: Crear una clase `Empleado`:**
    Escribe una clase `Empleado` que herede de la clase `Persona` y agregue un atributo `salario`.
84. **Sobrecarga de métodos:**
    Escribe una clase `Calculadora` con métodos sobrecargados para sumar dos números y tres números.
85. **Uso de getters y setters en una clase:**
    Escribe una clase `Coche` con atributos como `marca`, `modelo`, y `año`, y utiliza métodos getter y setter para acceder y modificar los valores.
86. **Polimorfismo con clases de animales:**
    Escribe una clase base `Animal` con un método `hablar()`. Luego, crea clases derivadas `Perro` y `Gato` que implementen su propia versión del método `hablar()`.
87. **Métodos estáticos en una clase:**
    Escribe una clase `Matematica` con un método estático que calcule el área de un círculo, dado su radio.
88. **Clase "Rectángulo" con métodos para área y perímetro:**
    Escribe una clase `Rectangulo` con métodos que calculen el área y el perímetro del rectángulo.
89. **Crear una clase "CuentaBancaria":**
    Escribe una clase `CuentaBancaria` que permita realizar depósitos, retiros y consultar el saldo.
90. **Manejo de excepciones en clases:**
    Escribe una clase `CuentaBancaria` que lance una excepción si el saldo es insuficiente para realizar un retiro.

Ejercicios sobre Manejo de Excepciones

91. **Manejo de división por cero:**
    Escribe un programa que solicite dos números al usuario y realice una división, manejando la excepción de una posible división por cero.
92. **Captura de error de tipo de dato:**
    Escribe un programa que reciba un número desde el usuario. Si el usuario ingresa un dato no numérico, el programa debe capturar la excepción y mostrar un mensaje de error adecuado.
93. **Manejo de acceso a índice fuera de rango en listas:**
    Escribe un programa que reciba una lista de números y permita al usuario acceder a un índice. Si el índice está fuera del rango, captura la excepción y muestra un mensaje.

94. **Crear una excepción personalizada:**
    Escribe una clase de excepción personalizada que sea lanzada cuando se ingrese un valor negativo en una lista de edades.
95. **Leer archivo con manejo de excepciones:**
    Escribe un programa que intente abrir y leer un archivo. Si el archivo no existe, captura la excepción y muestra un mensaje indicando que el archivo no fue encontrado.
96. **Manejo de excepciones con múltiples bloques `except`:**
    Escribe un programa que intente convertir una cadena de texto a un número entero. Maneja las excepciones tanto para valores no numéricos como para valores fuera de rango.
97. **Verificar tipo de archivo antes de abrir:**
    Escribe un programa que intente abrir un archivo para escritura. Antes de hacerlo, verifica si el archivo es de tipo `.txt`. Si no lo es, lanza una excepción personalizada.
98. **Comprobar si una clave existe en un diccionario:**
    Escribe un programa que reciba un diccionario y permita al usuario buscar una clave. Si la clave no existe, lanza una excepción y muestra un mensaje personalizado.
99. **Validación de entrada de datos (edad):**
    Escribe un programa que pida al usuario su edad. Si el valor ingresado no es un número o es negativo, lanza una excepción y muestra un mensaje de error.
100. **Manejo de múltiples errores en el mismo bloque `try`:**
    Escribe un programa que reciba una lista de números, calcule su promedio y gestione excepciones para dividir por cero o realizar operaciones con datos no numéricos.

Ejercicios sobre Complejidad Computacional (Algoritmos Eficientes)

101. **Tiempo de ejecución de un algoritmo de búsqueda lineal:**
    Escribe un programa que mida el tiempo de ejecución de un algoritmo de búsqueda lineal en una lista grande.
102. **Comparar la eficiencia de diferentes algoritmos de ordenación:**
    Escribe un programa que ordene una lista usando los algoritmos de burbuja, selección e inserción, y compare el tiempo de ejecución de cada uno con listas de diferentes tamaños.
103. **Implementar un algoritmo de búsqueda binaria:**
    Escribe un programa que implemente la búsqueda binaria en una lista ordenada y calcule su complejidad temporal en el peor de los casos.
104. **Optimización de la suma de números en una lista:**
    Escribe un programa que calcule la suma de todos los elementos de una lista de números grandes de manera eficiente, comparando la complejidad de usar un bucle `for` y una comprensión de listas.
105. **Problema de la mochila:**
    Escribe un programa que resuelva el problema de la mochila utilizando programación dinámica. Dado un conjunto de elementos con peso y valor, determina qué elementos maximizarán el valor total sin exceder la capacidad de la mochila.

106. **Algoritmo de Dijkstra:**
Escribe un programa que implemente el algoritmo de Dijkstra para encontrar el camino más corto entre dos nodos en un grafo ponderado.
107. **Algoritmo de clasificación rápida (QuickSort):**
Escribe un programa que implemente el algoritmo de clasificación rápida (QuickSort) y mide su tiempo de ejecución en listas de diferentes tamaños.
108. **Algoritmo de clasificación por fusión (MergeSort):**
Escribe un programa que implemente el algoritmo de clasificación por fusión (MergeSort) y compare su eficiencia con otros algoritmos de ordenación.
109. **Análisis de la complejidad temporal de un algoritmo recursivo:**
Escribe un programa que implemente un algoritmo recursivo simple (como el cálculo del factorial o Fibonacci) y explique cómo afecta la complejidad temporal.
110. **Algoritmo de búsqueda de subsecuencias comunes más largas:**
Escribe un programa que encuentre la subsecuencia común más larga entre dos cadenas usando un enfoque de programación dinámica.

Ejercicios sobre Programación Funcional

111. **Usar `map()` para transformar una lista:**
Escribe un programa que utilice la función `map()` para transformar una lista de números multiplicando cada número por 2.
112. **Usar `filter()` para filtrar elementos de una lista:**
Escribe un programa que utilice la función `filter()` para filtrar todos los números impares de una lista.
113. **Usar `reduce()` para calcular la suma de una lista:**
Escribe un programa que utilice `reduce()` del módulo `functools` para calcular la suma de los números en una lista.
114. **Composición de funciones:**
Escribe un programa que defina dos funciones sencillas (por ejemplo, `doble` y `cuadrado`) y las componga para realizar primero un cuadrado y luego un doble sobre un número.
115. **Uso de funciones lambda:**
Escribe un programa que utilice una función `lambda` para calcular el cubo de un número.
116. **Usar `zip()` para combinar dos listas:**
Escribe un programa que utilice la función `zip()` para combinar dos listas de manera que los elementos correspondientes estén emparejados en tuplas.
117. **Aplicar `sorted()` con una función de comparación personalizada:**
Escribe un programa que utilice `sorted()` con una función de comparación personalizada para ordenar una lista de cadenas por su longitud.
118. **Generadores para obtener una secuencia de Fibonacci:**
Escribe un generador que devuelva los números de la secuencia de Fibonacci de manera infinita y utiliza un ciclo para imprimir los primeros 10 números.

119. **Usar `any()` y `all()` para verificar condiciones en una lista:**
Escribe un programa que utilice las funciones `any()` y `all()` para verificar si hay algún número par en una lista o si todos los números son positivos.
120. **Funciones recursivas para calcular la suma de una lista:**
Escribe una función recursiva que calcule la suma de una lista de números sin usar ciclos explícitos.

Ejercicios sobre Algoritmos de Grafos

121. **Representar un grafo como lista de adyacencia:**
Escribe un programa que represente un grafo como lista de adyacencia y lo imprima en pantalla.
122. **DFS (Búsqueda en profundidad) en un grafo:**
Escribe un programa que implemente el algoritmo de búsqueda en profundidad (DFS) en un grafo representado por una lista de adyacencia.
123. **BFS (Búsqueda en anchura) en un grafo:**
Escribe un programa que implemente el algoritmo de búsqueda en anchura (BFS) en un grafo representado por una lista de adyacencia.
124. **Encontrar el camino más corto usando BFS:**
Escribe un programa que utilice el algoritmo BFS para encontrar el camino más corto entre dos nodos en un grafo no ponderado.
125. **Verificar si un grafo tiene un ciclo:**
Escribe un programa que utilice DFS para verificar si un grafo dirigido tiene un ciclo.
126. **Encontrar componentes conexas en un grafo no dirigido:**
Escribe un programa que utilice BFS o DFS para encontrar todas las componentes conexas en un grafo no dirigido.
127. **Algoritmo de Kruskal para encontrar el árbol de expansión mínima:**
Escribe un programa que implemente el algoritmo de Kruskal para encontrar el árbol de expansión mínima de un grafo ponderado.
128. **Algoritmo de Prim para encontrar el árbol de expansión mínima:**
Escribe un programa que implemente el algoritmo de Prim para encontrar el árbol de expansión mínima de un grafo ponderado.
129. **Topología de un grafo dirigido:**
Escribe un programa que utilice el algoritmo de ordenación topológica para encontrar un orden en el que los nodos de un grafo dirigido acíclico puedan ser procesados.
130. **Algoritmo de Floyd-Warshall para encontrar caminos más cortos:**
Escribe un programa que implemente el algoritmo de Floyd-Warshall para encontrar los caminos más cortos entre todos los pares de nodos en un grafo ponderado.