Intro. a Ciencia de la Computación

Ejercicios
Pregrado
2019-I
Ciencia de la Computación
Laboratorio

1. Elabore un programa que solicite al usuario el ingreso de números enteros positivos. El ingreso de números termina cuando el usuario ingresa -1.
   El programa tiene que almacenar en una primera lista los números ingresados que sean múltiplos de 5.

   A continuación, el programa solicita el ingreso de palabras. El ingreso de palabras termina cuando se ingresa la palabra: "FIN". El programa tiene que almacenar las palabras que tengan 3 ó más caracteres en una segunda lista. Ya sea que la palabra haya sido ingresada en mayúscula o minúscula, se almacena con mayúsculas en la segunda lista.

   Utilice el algoritmo de ordenamiento Insertion Sort para ordenar la primera lista de números de menor a mayor y el mismo algoritmo para ordenar las palabras de la segunda lista en orden alfabético (A-Z).

   Algunos ejemplos de diálogo de este programa serían:

```
Ingrese n mero: 55
Ingrese n mero: 12
Ingrese n mero: 35
Ingrese n mero: 11
Ingrese n mero: 90
Ingrese n mero: 25
Ingrese n mero: 88
Ingrese n mero: -1
---------------
Ingrese palabra: sol
Ingrese palabra: su
Ingrese palabra: comida
Ingrese palabra: el
Ingrese palabra: fuego
Ingrese palabra: aire
Ingrese palabra: luz
Ingrese palabra: ok
Ingrese palabra: FIN
----------------
Lista ordenada n meros(ascendente): 25 35 55 90
Lista ordenada palabras(A-Z): AIRE COMIDA FUEGO LUZ SOL
```

```
Ingrese n mero: 34
Ingrese n mero: 15
Ingrese n mero: 17
Ingrese n mero: 45
Ingrese n mero: 40
Ingrese n mero: 66
Ingrese n mero: -1
---------------
Ingrese palabra: colmillo
Ingrese palabra: vi
Ingrese palabra: lente
Ingrese palabra: campo
Ingrese palabra: cabeza
Ingrese palabra: la
Ingrese palabra: dos
Ingrese palabra: FIN
----------------
Lista ordenada n meros(ascendente): 15 40 45
Lista ordenada palabras(A-Z): CABEZA CAMPO COLMILLO DOS
   LENTE
```

2. Desarrolle un programa que permita el ingreso de los meses del año (validar que solo deben ingresar los meses de "enero" a "diciembre" en minúscula). Se termina de ingresar con "FIN" Utilizando el algoritmo de ordenamiento Insertion Sort, mostrar los meses en orden descendente. EJEMPLO

```
Mes: Enero
Error
Mes: FEBRERO
Error
Mes: enero
Mes: diciembre
Mes: marzo
Mes: FIN
Lista ordenada en descendente
marzo
diciembre
enero
```

3. El Instituto Nacional de Estadística e Informática (INEI) dio a conocer que de acuerdo con los Resultados de los Censos Nacionales 2017: XII de Población, VII de Vivienda y III de Comunidades Indígenas, ejecutados el 22 de octubre del año pasado, la población de los distrito de Lima es el siguiente:

```
inei = {"Molina":175237, "Callao":426649, "Victoria
   ":174958,
  "Comas":532403, "Agustino":194474, "Carabayllo":305963,
  "Olivos":377532, "Cieneguilla":47860, "Lince":51054,
  "Independencia":220654, "Jesus Maria":73439, "Ate
     ":638345,
  "Carmen Legua":43156, "Mi Peru":52722, "Ventanilla
     ":356040,
  "Lima":276861, "Ancon":43951, "Barranco":30698,
  "Bellavista":78489, "La Perla":64111, "Brena":77291,
  "Chaclacayo":44271, "Chorrillos":330483, "Lince":51054 }
```

Escribe un programa que realice lo siguiente:

- Ordenar la población de cada distrito en forma descendente y guardelo en una lista.
- Imprimir la lista ordenada.
- Imprimir la población más alta.
- Imprimir la población más baja.

Para obtener una lista con la población ordenada en forma descendente, implemente y use la siguiente función:

- `ordenar_poblacion` recibe como parámetro el `diccionario inei`, y utilizando el método `Insert Sort` retorne la lista de la población ordenada en forma descendente.

Restricciones: No puede utilizar las funciones `sort, min y max`.

El resultado de la ejecución de programa es el siguiente:

```
Poblacion ordenada: [638345, 532403, 426649, 377532,
   356040, 330483, 305963, 276861, 220654, 194474, 175237,
    174958, 78489, 77291, 73439, 64111, 52722, 51054,
   47860, 44271, 43951, 43156, 30698]
Poblacion mas alta: 638345
Poblacion mas baja: 30698
```

4. La investigadora Counterpoint Research presentó su informe sobre ventas de teléfonos móviles correspondientes al cierre de 2018, en el que se evidenció, por primera vez en una caída de 4% en las ventas de estos dispositivos móviles frente a la misma fecha en 2017, Las ventas por millones de unidades es la siguiente:

```
ventas = {"Xiaomi":96, "Oppo":119, "LG":55.9, "Lenovo
   ":49.9,
          "Nokia":7.7, "TecnoPro":13.1, "Dell":6.1,
          "IBM":0.6, "Sahito":34.6, "Samsung":318.1,
```

```
            "Apple":215.8, "Vivo":100.2, "Huawei":153.1,
            "Aio":45.6, "FirePhone": 29.7, "Motorola":20.3 }
```

Escribe un programa que realice lo siguiente:

- Ordenar las ventas de cada empresa en forma ascendente y guardelo en una lista.
- Imprimir la lista ordenada.
- Imprimir la venta más baja.
- Imprimir la venta más alta.

Para obtener una lista ordenada de ventas en forma ascendente, implemente y use la siguiente función:

- `ordenar_ventas` recibe como parámetro el diccionario `ventas`, y utilizando el método `Insert Sort` retorne la lista ordenada de ventas en forma ascendente.

Restricciones: No puede utilizar las funciones `sort, min y max`.

El resultado de la ejecución de programa es el siguiente:

```
Ventas ordenada: [0.6, 6.1, 7.7, 13.1, 20.3, 29.7, 34.6,
   45.6, 49.9, 55.9, 96, 100.2, 119, 153.1, 215.8, 318.1]
Venta mas baja: 0.6
Venta mas alta: 318.1
```

5. Dada la siguiente matriz de números compuestos:

```
compues = [[8, 9, 14, 15, 16, 18, 4, 6, 10, 12 ],
           [21, 22, 25, 26, 27, 20, 24],
           [34, 28, 32, 33, 30, ],
           [38, 39, 42, 44, 35, 36, 40],
           [52, 54, 45, 48, 49, 50, 51, 46 ]]
```

Escribe un programa que realice lo siguiente:

- Ordenar los números de cada lista en forma ascendente
- Imprimir las listas ordenadas.
- Imprimir el máximo número de cada lista

Para ordenar los números de cada lista en forma ascendente, implemente y use la siguiente función:

- `ordenar_matriz` recibe como parámetro la matriz`compues`, y utilizando el método `Insert Sort` ordenada las listas en forma ascendente.

Restricciones: No puede utilizar las funciones `sort, min y max`.

El resultado de la ejecución de programa es el siguiente:

```
[4, 6, 8, 9, 10, 12, 14, 15, 16, 18]
[20, 21, 22, 24, 25, 26, 27]
[28, 30, 32, 33, 34]
[35, 36, 38, 39, 40, 42, 44]
[45, 46, 48, 49, 50, 51, 52, 54]
18
27
34
44
54
```

6. Se quiere hacer un análisis de las aplicaciones móviles más populares por algunos estudiantes. El objetivo es mostrar las aplicaciones más usadas desde las más populares hasta las menos populares. Escribe una función en Python que reciba una lista de aplicaciones y su ranking, representado por un número, y retorne el nombre de las aplicaciones más populares de forma decreciente.

   Por ejemplo:

```
Productos: [("Tinder", 100), ("Gmail", 20), ("Glovo", 50)]
Tinder Glovo Gmail
```

```
Productos: [("WhatsApp", 100), ("Instagram", 100)]
WhatsApp Instagram
```

   Consideraciones: No se considerará válido el uso de la función sort() de python o de alguna otra función que modifique listas como append() o pop(), sin embargo se puede usar la función len(). Además su algoritmo no deberá hacer uso de alguna lista o diccionario adicional al parámetro de la función.

7. El Señor Demetrio viajó a Brasil para apoyar a la selección peruana en la Copa América 2019 por un intervalo de tiempo no definido, dejando en su casa una serie de deudas ocasionadas por los fuertes gastos de dicho viaje. A su regreso, él decide trabajar duro para amortiguar dichas deudas empezando desde la mayor hasta la menor. Para ello, se le pide a usted ayudar a don Demetrio diseñando una función que ordene las deudas de forma descendente. Las deudas estan almacenadas en una lista de tuplas.

   Plantilla del código:

```
def ordenarDeudas(deudas):
      #
      # implemente la funcion de ordenacion aqui
      #
```

```
deudas=[("luz",115),("agua",50),
        ("banco",3000),("colegio",850)]
print("Deudas ordenadas:", ordenarDeudas(deudas))
```

Salida:

```
Deudas ordenadas: [("banco",3000),("colegio",850),
                   ("luz",115),("agua",50)]
```

Restricciones: No se considerará válido el uso de la función sort() de python o de alguna otra función que modifique listas como append() o pop(), sin embargo se puede usar la función len(). Además su algoritmo no deberá hacer uso de alguna lista o diccionario adicional al parámetro de la función.

8. Ordenar los siguientes números de menor a mayor usando cualquiera de los algoritmos estudiados en clase.

```
n meros = [10, 30, 110, 110.2, 20, 50, 2, 900, 2,13]
```

9. **Estrellas:** Genere una tabla de la siguiente lista de estrellas y sus propiedades ordenadas por luminosidad (L), donde $L = T^4 * R^2$, siendo T la temperatura y R el radio de la estrella

estrella	**radio**	**temperatura**
UY-Scuti	1.71	3365
Canis Mayoris	1.42	3490
KY Cyg	1.42	3500
Mu Cephei	1.26	3750
S Persei	1.20	3000
VX Sagittarii	1.12	3000

```
Ordenado por luminosidad:
VX Sagittarii , S Persei , UY-Scuti , Canis Mayoris , KY
   Cyg , Mu Cephei
```

10. Crea una función que reciba una lista, la ordene de acuerdo a alguno de los metodos de ordenamiento vistos en clase, y también indique cuantos movimientos en la lista tuvo que realizar.

11. Crea una función que reciba una lista cuyo contenido sean números enteros y que ordene la lista de mayor a menor, utilizando una de las técnicas de ordenamiento aprendidas.

12. Crea una función que reciba dos listas cuyo contenido sean números enteros y que te devuelva una lista con todos los elementos de las dos listas, ordenados de menor a mayor, usando alguno de los algoritmos de ordenamiento aprendidos en clase.

13. Dado un conjunto de números leidos del archivo: `"numeros.txt"`

```
14 25
23 12
45 56
23 45
12 34
90 12
```

- Se pide leer los datos del archivo y convertirlos en un array llamado `numeros`
- Se pide ordenar los numeros de forma `ascendente` usando Insertion Sort.

Un ejemplo de diálogo de este programa sería:

```
numeros = [14, 25, 23, 12, 45, 56, 23, 45, 12, 34, 90, 12]
insertionsort(numeros)
print(numeros)
[12, 12, 12, 14, 23, 23, 25, 34, 45, 45, 56, 90]
```

14. Diseñe e implemente una función que permita hallar el segundo mayor elemento de una lista de números. Considere los siguientes ejemplos:

```
test = [5, 4, 1, 2, 8, 3, 2, 9]
output = 8

test = [4, 1, 9, 3, 10, 6, 4, 3]
output = 9

test = [8, 3, 4, 8, 2, 1, 6, 7]
output = 8
```

Considerar las siguientes restricciones:

- No se debe utilizar sorted ni sort (si requiere ordenar la lista, deberá implementar la función).
- No se deben crear listas adicionales.
- No se debe utilizar métodos que alteren el tamaño de la lista (remove, slicing, pop).
- El tamaño minimo de la lista será 2.

15. Diseñe e implemente una función que permita hallar el segundo menor elemento de una lista de números. Considere los siguientes ejemplos:

```
test = [5, 4, 1, 2, 8, 3, 2, 9]
output = 2

test = [4, 1, 9, 3, 10, 6, 4, 3]
```

```
output = 3

test = [8, 3, 4, 8, 2, 1, 6, 7]
output = 2
```

Considerar las siguientes restricciones:

- No se debe utilizar sorted ni sort (si requiere ordenar la lista, deberá implementar la función).
- No se deben crear listas adicionales.
- No se debe utilizar métodos que alteren el tamaño de la lista (remove, slicing, pop).
- El tamaño minimo de la lista será 2.

16. La empresa de juegos mecánicos "Divertilandia" desea usar un programa en Python para su nueva atracción "Montaña rusa" la cual tiene 5 distintas filas. Cada fila está determinada para determinado rango de altura.

    Los rangos de las filas son los siguientes:

    - Fila 1: 50-89 cm
    - Fila 2: 90- 119 cm
    - Fila 3: 120 – 160 cm
    - Fila 4: 160 – 179 cm
    - Fila 5: 180 – 200 cm

    Crear un programa que cumpla las siguientes condiciones:

    - El usuario debe ingresar 5 números los cuales representan la altura en centímetros.
    - El mínimo de altura es 50 y el máximo 200. No se debe admitir un número que esté fuera del rango.
    - Generar una función que ordene la lista utilizado el método de inserción de mayor a menor
    - Generar una función que genere e imprima un diccionario donde se indiquen las filas y los números ingresados correspondientes a cada una de ellas.
    - Tener en cuenta que puede haber filas vacías

    Ejemplo:

```
Altura 1: 125
Altura 2:  190
Altura 3: 70
Altura 4:  95
Altura 5: 150
Resultado:  {'Fila 1':'70', 'Fila 2':'95', 'Fila 3':'125', '
   Fila 3','150', 'Fila 5','190'}
```

17. La empresa de juegos mecánicos "Divertilandia" desea usar un programa en Python para su nueva atracción "Montaña rusa" la cual tiene 5 distintas filas. Cada fila está determinada para determinado rango de altura.

    Los rangos de las filas son los siguientes:

    - Fila 1: 50-89 cm
    - Fila 2: 90- 119 cm
    - Fila 3: 120 – 160 cm
    - Fila 4: 160 – 179 cm
    - Fila 5: 180 – 200 cm

    Crear un programa que cumpla las siguientes condiciones:

    - El usuario debe ingresar 5 números los cuales representan la altura en centímetros.
    - El mínimo de altura es 50 y el máximo 200. No se debe admitir un número que esté fuera del rango.
    - Generar una función que ordene la lista utilizado el método de inserción de mayor a menor
    - Generar una función que genere e imprima un diccionario donde se indiquen las filas y los números ingresados correspondientes a cada una de ellas.
    - Tener en cuenta que puede haber filas vacías

    Ejemplo:

```
Altura 1: 125
Altura 2:  190
Altura 3: 70
Altura 4:  95
Altura 5: 150
Resultado:  {'Fila 1':'70', 'Fila 2':'95', 'Fila 3':'125', '
   Fila 3','150', 'Fila 5','190'}
```

18. Diseñar y crear un programa que genere una lista de datos de tamaño n con valores entre 1 y 99 generados aleatoriamente y que ordene solo los elementos en la posiciones impares de la lista.

    NOTA: El programa no debe utilizar la función sort de python, ni listas, ni diccionarios adicionales o estructuras similares.

```
    Ingrese el numero: 6
    La lista original es: [22, 3, 11, 9, 7, 2]
    La lista ordenada es: [22, 2, 11, 3, 7, 9]
```

```
    Ingrese el numero: 8
    La lista original es: [10, 1, 11, 8, 14, 4, 17, 5]
    La lista ordenada es: [10, 1, 11, 4, 14, 5, 17, 8]
```

19. Diseñar y crear un programa que genere una lista de datos de tamaño n con valores entre 1 y 99 generados aleatoriamente y que ordene solo los elementos en la posiciones multiplo de 3 (incluida la posicion 0) de la lista.

    NOTA: El programa no debe utilizar la función sort de python, ni listas, ni diccionarios adicionales o estructuras similares.

```
    Ingrese el numero: 9
    La lista original es: [5, 10, 14, 1, 11, 12, 8, 20, 10]
    La lista ordenada es: [1, 10, 14, 5, 11, 12, 8, 20, 10]
```

```
    Ingrese el numero: 10
    La lista original es: [20, 1, 4, 30, 14, 2, 41, 5, 7, 10]
    La lista ordenada es: [10, 1, 4, 20, 14, 2, 30, 5, 7, 41]
```

20. Utilice un algoritmo de ordenación descrito en clases para poder ordenar las listas de menor a mayor en el diccionario generado por el programa.

```
n = int(input("ingrese n:"))
m = int(input("ingrese m:"))
d = {}
from random import randrange
for i in range(1, m+1):
  lista = [randrange(1, n) for x in range(1,n+1)]
  d[i] = lista

#aqui completar el programa

print(d)
```

    Input Format

    Ingreso de 2 números n y m.

    Veamos algunos ejemplos:

```
input:
5
3

output:
{1: [1, 1, 1, 1, 3], 2: [1, 3, 3, 5, 5], 3: [1, 2, 3,4, 5]}
```

```
input:
20
3
output:
{1: [1, 1, 1, 2, 2, 2, 3, 3, 4, 6, 7, 11, 12, 12, 15,16, 18,
    19, 20, 20], 2: [2, 3, 4, 5, 7, 7, 7, 9, 10, 12, 12, 13,
    13, 15, 16, 17,17, 17, 18, 18], 3: [1, 1,2, 4, 5, 7, 10,
    12, 13, 14, 14, 14, 14, 15, 18, 18, 19, 19, 19, 20]}
```

21. Utilice un algoritmo de ordenación descrito en clases para poder ordenar de menor a mayor las listas en la matriz generada por el programa.

```
n = int(input("ingrese n:"))
m = int(input("ingrese m:"))
matriz = []
from random import randrange
for i in range(1, m+1):
  lista = [randrange(1, n) for x in range(1,n+1)]
  matriz.append(lista)

#aqui completar el programa

print(matriz)
```

Input Format

Ingreso de 2 números n y m.

Veamos algunos ejemplos:

```
input:
5
3

output:
[[1, 1, 1, 1, 3],[1, 3, 3, 5, 5],[1, 2, 3, 4, 5]]
```

```
input:
20
3
output:
[[1, 1, 1, 2, 2, 2, 3, 3, 4, 6, 7, 11, 12, 12, 15,16, 18,
   19, 20, 20], [2, 3, 4, 5, 7, 7, 7, 9, 10, 12, 12, 13, 13,
    15, 16, 17,17, 17, 18, 18], [1, 1,2, 4, 5, 7, 10, 12,
   13, 14, 14, 14, 14, 15, 18, 18, 19, 19, 19, 20]]
```

22. Elabore un programa que permita al usuario ingresar un número entero positivo N. Valide que sea un número positivo mayor o igual a 5.
Luego ingresará N números enteros a una lista ordenada. El programa imprime la lista. A continuación, el usuario ingresa un número a buscar en la lista. Asuma que el usuario ingresa un número entero positivo.

    Si encuentra el número, nos informa que lo encontró y si no lo encontró, crea e imprima una nueva lista con la misma cantidad de elementos que la otra lista, pero donde cada elemento es igual a cero.

    Utilice el algoritmo de Búsqueda Binaria para encontrar el número buscado.

    Algunos ejemplos de diálogo:

```
Ingrese un n mero entero: -3
Ingrese un n mero entero: -343
Ingrese un n mero entero: 5
11
44
66
87
90
Lista: [11, 44, 66, 87, 90]
Ingrese n mero a buscar: 33
No se encontr  33.  -> [0, 0, 0, 0, 0]
```

```
Ingrese un n mero entero: -4
Ingrese un n mero entero: 6
23
78
134
345
444
Lista: [23, 78, 134, 345, 444]
Ingrese n mero a buscar: 345
Se encontr  345
```

23. Elabore un programa que pida el nombre de usuario y clave desde teclado. Si el cambio es válido, se muestra el mensaje CORRECTO, en caso contrario se muestra el mensaje ERROR. El programa debe realizar las siguientes validaciones - No se debe permitir ingresar una clave anteriormente usada

    EJEMPLO

```
Usuario: admin
Clave : utec
CORRECTO
Desea cambiar clave[S/N]: S
```

```
Usuario: admin
Clave: 1234
CORRECTO
Desea cambiar clave[S/N]: S
Usuario: admin
Clave: utec
ERROR
Desea cambiar clave[S/N]: N
FIN
```

24. Escribe un programa que realice lo siguiente:

    - Generar la siguiente lista de números ordenados:

```
numeros = [15, 30, 45, 60, 75, 90, 105, 120, 135, 150,
           165, 180, 195, 210, 225, 240, 255, 270, 285,
           300, 315, 330, 345, 360, 375, 390, 405, 420,
           435, 450, 465, 480, 495, 510, 525, 540, 555,
           570, 585, 600, 615, 630, 645, 660, 675, 690,
           705, 720, 735, 750, 765, 780, 795, 810, 825,
           840, 855, 870, 885, 900, 915, 930, 945, 960,
           975, 990]
```

    - Solicitar al usuario que ingrese un número.
    - Buscar el número ingresado en la lista
    - Imprime "`El número está en la posición X de la lista`" cuando el número este en la lista y "`El número NO está en la lista`" en caso contrario.

    Para buscar el número ingresado en la lista, implemente y use la siguiente función:

    - `buscar_numero` recibe como parámetro una `lista` y el número a buscar, y utilizando el algoritmo de `Búsqueda Binaria` retorne la posición del número buscado, en caso de no estar el elementos retornar -1.

    Algunos ejemplos de diálogo son:

```
Ingrese numero a buscar: 645
El n mero  est   en la posici n  42 de la lista
```

```
Ingrese numero a buscar: 153
El n mero NO est  en la lista
```

25. Escribe un programa que realice lo siguiente:

    - Generar la siguiente lista de números múltiplos de 7:

```
numeros = [7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77,
           84, 91, 98, 105, 112, 119, 126, 133, 140,
           147, 154, 161, 168, 175, 182, 189, 196, 203,
           210, 217, 224, 231, 238, 245, 252, 259, 266,
           273, 280, 287, 294, 301, 308, 315, 322, 329,
           336, 343, 350, 357, 364, 371, 378, 385, 392,
           399, 406, 413, 420, 427, 434, 441, 448, 455,
           462, 469, 476, 483, 490, 497]
```

- Solicitar al usuario que ingrese un número.
- Buscar el número ingresado en la lista de números
- Imprime "`El número está en la posición X de la lista`" cuando el número este en la lista y "`El número NO está en la lista`" cuando no este en la lista.

Para buscar el número ingresado en la lista, implemente y use la siguiente función:

- `buscar_numero` recibe como parámetro una `lista` y el número a buscar, y utilizando el algoritmo de `Búsqueda Binaria` retorne la posición del número buscado, en caso de no estar el elementos retornar -1.

Algunos ejemplos de diálogo son:

```
Ingrese numero a buscar: 125
El n mero NO est  en la lista
```

```
Ingrese numero a buscar: 385
El n mero est  en la posici n  54 de la lista
```

26. Escribe un programa que realice lo siguiente:

- Generar la siguiente lista de números múltiplos de 7:

```
numeros = [-99, -88, -77, -66, -55, -44, -33, -22,
           -11, 0, 11, 22, 33, 44, 55, 66, 77, 88, 99,
           110, 121, 132, 143, 154, 165, 176, 187, 198,
           209, 220, 231, 242, 253, 264, 275, 286, 297]
```

- Solicitar al usuario que ingrese un número.
- Buscar el número ingresado en la lista de números
- Imprime "`El número está en la posición X de la lista`" cuando el número este en la lista y "`El número NO está en la lista`" cuando no este en la lista.

Para buscar el número ingresado en la lista, implemente y use la siguiente función:

- `buscar_numero` recibe como parámetro una `lista` y el número a buscar, y utilizando el algoritmo de `Búsqueda Binaria` retorne la posición del número buscado, en caso de no estar el elementos retornar -1.

Algunos ejemplos de diálogo son:

```
Ingrese numero a buscar: 333
El n mero NO est  en la lista
```

```
Ingrese numero a buscar: 253
El n mero  est  en la posici n  32 de la lista
```

27. Escribe una función en Python que permita saber si un número se encuentra en una lista ordenada. Si el número a buscar se encuentra o no en la lista, entonces se deberá mostrar un mensaje apropiado, Adicionalmente, deberá contar el número de comparaciones hechas por su algoritmo. El número total de comparaciones debe ser menor al tamaño de la lista de números y puede asumir que la lista tiene como mínimo 5 elementos.

    Por ejemplo:

```
Lista de numeros: [1,2,3,4,5]
Numero a buscar: 10
El numero 10 no se encuentra despues de 3 comparacion(es)
```

```
Lista de palabras: [10, 20, 30, 40, 50, 60, 70, 80, 90]
Numero a buscar: 50
El numero 50 se encuentra despues de 1 comparacion(es)
```

28. Se plantea el siguiente juego: en una mesa se tiene una fila de n vasos y debajo de cada vaso hay una bolita enumerada, se pide hallar la mínima cantidad de vasos que necesito levantar para verificar la existencia de un número dado. Asumir que las bolas estan ordenadas de forma descendente y usted tiene que aplicar un algoritmo de búsqueda que minimize el número de vasos a levantar.

    Plantilla del código:

```
def contarVasosLevantados(lista, buscar):
        #
        # implemente la funcion de busqueda aqui
        #

bolitas = [21,18,15,11,9,8,5,2]
buscar = 5
conteo = contarVasosLevantados(bolitas, buscar)
print("Vasos levantados:", conteo)
```

Salida:

```
Vasos levantados: 3
```

29. Escribir un programa que realize la búsqueda de un número entero dentro de una matriz de tamaño $m \times n$. La matriz tiene las siguientes propiedades:

    - Los enteros de cada fila están ordenados de izquierda a derecha.
    - El primer entero de cada fila es mayor que el último entero de la fila anterior.

```
Ejemplo de matriz en Python:
A=[[1,2,3,4],[5,6,7,8],[9,10,11,12]]
```

    Indicaciones:

    - Usar **búsqueda binaria**.
    - El usuario ingresa el numero que desea buscar en la matriz.
    - Imprimir la ubicación (fila,columna) en caso el elemento fuera encontrado.
    - Caso contrario, mostrar el mensaje "El número no existe en la matriz".

30. **Sistema Solar:** crear un código que realize una búsqueda de datos del Sistema Solar, basado en la tabla e imprima los siguientes resultados:

planeta	**diámetro**	**año**	**día**
Mercurio	4878	88	59
Venus	12104	225	5784
Tierra	12760	365	24
Marte	6787	687	24
Jupiter	139822	4343	10
Saturno	120500	10767	11
Urano	51120	30660	18
Neptuno	49530	60225	19

    - planetas con di'ametro mayor que la Tierra
    - planetas con años más largos que la Tierra
    - planetas con diámetro menor que la Tierra y días iguales o más cortos que los de la Tierra

```
planetas con di\'ametro mayor que la Tierra: J\'upiter,
   Saturno, Urano, Neptuno
planetas con a\~nos m\'as largos que la Tierra: Marte, J\'
   upiter, Saturno, Urano, Neptuno
planetas con di\'ametro menor que la Tierra y d\'ias iguales
    o m\'as cortos que los de la Tierra: Marte
```

31. Crea una función que reciba una lista ordenada con 100 números y un número a buscar, y luego por medio de búsqueda binaria encuentre la posición de dicho número. Además deberá devolver cuántas búsquedas tuvo que realizar para hallar el resultado.

32. Implemente el algoritmo de busqueda binaria de forma recursiva.

    - El algoritmo debe retornar el indice en caso encuentre el elemento a buscar.
    - El algoritmo debe retornar -1 en caso no se encuentre el elemento a buscar.

    Un ejemplo de diálogo de este programa sería:

```
lista = [5, 3, 4, 1, 2]
elemento_buscar(4)
3
```

```
lista = [5, 3, 4, 1, 2]
elemento_buscar(6)
-1
```

33. Implemente una función que realice el algoritmo "búsqueda binaria" de manera recursiva para determinar si un número está o no en una lista de números. Asuma que la lista que ingresará a la función se encuentra ordenada de **mayor a menor**.

34. Implemente una función que realice el algoritmo "búsqueda binaria" de manera iterativa para determinar si un número está o no en una lista de números. Asuma que la lista que ingresará a la función se encuentra ordenada de **mayor a menor**.

35. Existe el requerimiento de programar una aplicación móvil de magia en la cual se encuentra el truco "Adivina las cartas" por lo que desea crear un programa en python que le permite ordenar 52 cartas.

    Las cartas están ordenadas de manera que el primer mazo es corazones, el segundo espadas, el tercero cocos y el cuarto trebol.

    Generar un programa en python que cumpla con lo siguiente:

    - El usuario debe ingresar dos números e indicar el mazo (Ejemplos de input: 'K, espadas', 'J,trebol')
    - Crear una función que genere el diccionario de cartas respetando el orden de mazo propuesto en el enunciado.
    - Crear otra función que reciba el input del usuario, busque en forma binaria las posiciones y devuelva como resultado un solo string donde la posición 1 (que corresponde al input 1) es seguida de la posición 2 (que corresponde al input 2).

    Ejemplo:

```
Carta 1:  '10,corazones'
Carta 2: 'Q,treboles'
Resultado: 1051
```

36. Existe el requerimiento de programar una aplicación móvil de magia en la cual se encuentra el truco "Adivina las cartas" por lo que desea crear un programa en python que le permite ordenar 52 cartas.

    Las cartas están ordenadas de manera que el primer mazo es corazones, el segundo espadas, el tercero cocos y el cuarto trebol.

    Generar un programa en python que cumpla con lo siguiente:

    - El usuario debe ingresar dos números e indicar el mazo (Ejemplos de input: 'K, espadas', 'J,trebol')
    - Crear una función que genere el diccionario de cartas respetando el orden de mazo propuesto en el enunciado.
    - Crear otra función que reciba el input del usuario, busque en forma binaria las posiciones y devuelva como resultado un solo string donde la posición 1 (que corresponde al input 1) es seguida de la posición 2 (que corresponde al input 2).

    Ejemplo:

```
Carta 1:  '10,corazones'
Carta 2: 'Q,treboles'
Resultado: 1051
```

37. Diseñar y crear una programa que lea un archivo, el cual incluirá en cada fila un identificador de pais (expresado por el código del pais basado en el estandar ISO 3166-1), seguido por la tasa de crecimiento poblacional (expresado en %). El programa deberá de generar con la información un diccionario, donde la clave sea el código del pais y el valor la tasa de crecimiento poblacional y calcular el promedio de la tasa de crecimiento poblacional.

    Contenido de archivo Datos.txt :

```
PE  1.29
JP -0.10
AR  1.03
US  0.77
BR  0.92
IN  1.28
CN  0.51
CL  0.91
```

    La ejecución sería :

```
Ingrese el nombre del archivo: Datos.txt
La Tasa de Crecimiento Poblacional promedio es: 0.826
```

38. Diseñar y crear una programa que lea un archivo, el cual incluirá en cada fila un identificador de pais (expresado por el código del pais basado en el estandar ISO 3166-1),

seguido por la tasa de crecimiento poblacional (expresado en %). El programa deberá de generar con la información un diccionario, donde la clave sea el código del pais y el valor la tasa de crecimiento poblacional y presentar el código del país con menor tasa poblacional.

Contenido de archivo Datos.txt :

```
PE  1.29
JP -0.10
AR  1.03
US  0.77
BR  0.92
IN  1.28
CN  0.51
CL  0.91
```

El resultado seria :

```
Ingrese el nombre del archivo: Datos.txt
El pais con menor tasa poblacional es: JP
```

39. Utilice un algoritmo de busqueda lineal para encontrar todos los numeros mayores a un X valor en una lista de N valores aleatorios.

```
from random import randrange
n = int(input("numeros aleatorios a generar:"))
lista = [randrange(1, 101) for x in range(1,n+1)]

numerosEncontrados = []
x = int(input("ingrese el numero a buscar:"))

# ToDO
print(lista)
print(numerosEncontrados)
```

```
input:
14
21
output:
[59, 51, 10, 35, 19, 58, 61, 20, 50, 35, 62, 21, 34, 11]
[59, 51, 35, 58, 61, 50, 35, 62, 34]
```