	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Jesus Cruz Navarro
Asignatura:	Estructura de Datos y Algoritmos II
Grupo:	Grupo 1
No de Práctica(s):	Práctica 5 – Algoritmos de búsqueda parte 2
Integrante(s):	Edwin Jaret Santiago Díaz
No. de Equipo de cómputo empleado:	22
No. de Lista o Brigada:	22
Semestre:	2022 - 2
Fecha de entrega:	20 marzo 2022
Observaciones:	

CALIFICACIÓN: _____

Algoritmos de búsqueda parte 2

Objetivos

1. Implementar una tabla hash usando chaining para el manejo de colisiones y compararlo con otro método de búsqueda.
2. Desarrollar un programa que simule un sistema de login utilizando tabal hash.

Desarrollo

Programa 1:

En el programa “Database.py” se implementó una Tabla Hash para guardar 10,000 elementos. En esta tabla hash se guardan objetos de la clase Usuario en los cuales contienen el nombre completo del usuario, el nombre del usuario y su contraseña. Estos datos se obtienen del programa “Database_12k.py”.

La Tabla Hash guarda los elementos con el método chaining en donde la clave es el nombre del usuario y conforme al número que devuelve este método se guardará en el dicho índice.

El método chaining consiste en la suma del valor en código ASCII de cada carácter del nombre completo del usuario y esta suma será dividida entre el tamaño de la Tabla Hash (10,000), lo que retorna el método es el módulo de esta división.

Se implementó un sistema en donde se le solicita al usuario que ingrese el nombre completo de un usuario y con su contraseña, el programa buscará a través del método *buscar* el nombre de usuario y validará si este existe, una vez encontrado validará la contraseña registrada en el sistema con la contraseña ingresada por el usuario. Se le notifica al usuario en caso de que el usuario no se encuentra en el sistema, la contraseña es errónea o la bienvenida al sistema cuando la contraseña y el usuario son correctos.

Se mostrará a continuación el funcionamiento del programa al ingresar 3 usuarios que existen con sus respectivas contraseñas, cuando el nombre del usuario no existe y cuando la contraseña es errónea.

Nombre del usuario y contraseñas correctas:

```
Cual es el nombre del usuario? Lyndsey Aarons
Cual es la contraseña? wKqNf2

Acceso Autorizado: Bienvenido Lyndsey Aarons
```

```
Cual es el nombre del usuario? Rodrigo Curneen
Cual es la contraseña? mPeL5GmA

Acceso Autorizado: Bienvenido Rodrigo Curneen
```

```
Cual es el nombre del usuario? Hannie Massimi
Cual es la contraseña? A4MZdQ

Acceso Autorizado: Bienvenido Hannie Massimi
```

Nombre del usuario no encontrado:

```
Cual es el nombre del usuario? Lyndsey Aaron
Cual es la contraseña? wKqNf2

Usuario no encontrado
```

Contraseña del usuario incorrecta:

```
Cual es el nombre del usuario? Lyndsey Aarons
Cual es la contraseña? wKqNf

Acceso Denegado: Contraseña Incorrecta
```

Programa 2:

Se analiza el tiempo que el programa tarda en buscar 10 usuarios (son aleatoriamente escogidos por el programa) por el método buscar de la Tabla Hash y por la búsqueda lineal. Estos son los resultados:

```
Estos son los usuarios a buscar:
        Reggis Coronas
        Wendie Sotheby
        Selia O'Carroll
        Huntley Hoppner
        Pete Skade
        Colly Dodwell
        Kathrine Gelland
        Aura Bonwick
        Steffi Spilling
        Emanuel Rouby

Tiempos de ejecución PROMEDIO (10 ejecuciones)
       Hash Table:  6.750004831701517e-05
       Linear:  0.005123100010678172
```

Nos muestra que a través del método *buscar* toma menos tiempo que buscar a través de la búsqueda lineal.

El *Load Factor* de la tabla hash es de **0.118**

Conclusiones

Una Tabla Hash es una estructura de datos que asocia llaves con valores. Su principal característica es su búsqueda de manera eficiente en donde permite el acceso a elementos almacenados a partir de una clave generada. La Tabla Hash transforma una clave con el método Chaining en donde se identifica la posición donde se guardará un valor deseado.

Las Tabla Hash pueden guardar mucha información de manera eficiente (en este caso, se guardó 10,000 elementos) en donde buscar o agregar un elemento no requiere de muchos recursos ni requiere de mucho tiempo, en cambio si se utiliza un arreglo requeriría muchos recursos.

El factor de carga de la tabla creada es de 0.1, menor que 0.7, dando a entender que la probabilidad de colisiones es muy baja.