**Obligatorio 2021 Programación 2**

Parte 1:

1. Considerar una pequeña red social de 100000 usuarios. De cada uno de ellos se conoce nombre, apellido, cedula identidad, correo electrónico y usuario y contraseña.
2. Cada usuario puede ser amigo de cualquiera de esos usuarios en la red.
3. Tiene posteos de su propia página.
4. Interesa dado un usuario devolver la lista de sus amigos y de sus posteos.
5. También interesa llevar un control de la cantidad de usuarios.

Parte 2: Ahora la red crece y no se sabe cuál puede ser el límite, adaptar el análisis a la nueva situación.

Se pide:

Para las partes 1 y 2 realizar el TAD correspondiente

Programar en C++ las realidades (el programa contará con un menú). Se debe contemplar el alta, baja y modificación de los usuarios, de los posteos y los listados correspondientes.

Dado un usuario se podrá listar sus amigos y sus posteos. Debe programar un menú para atender los requerimientos.

Realizar el análisis de uno de los algoritmos realizados en cada una de las partes el cual se coordinará con el docente

Bienvenidos ---Red Social

1-Usuarios

1-Altas

Nombre:

Apellido:

Ci sin puntos ni guion:

Valida ci.

Correo electrónico:

Usuario:

Valida usuario.

Contraseña:

Confirma el ingreso S/N:

2-Bajas

Digite ci del usuario sin puntos ni guion:

Muestra datos.

3-Modificaciones

Digite ci del usuario sin puntos ni guion:

Usuario:

Nombre: muestra nombre

¿Desea modificar el nombre? S/N

Pide nombre.

Apellido: muestra apellido

¿Desea modificar el apellido? S/N

Pide apellido.

Ci: muestra ci

¿Desea modificar la ci? S/N

Pide ci

Correo electrónico: muestra Correo electrónico

¿Desea modificar Correo electrónico? S/N

Pide ce.

Usuario: muestra Usuario

¿Desea modificar Usuario? S/N

Pide usuario.

Contraseña: muestra Contraseña

¿Desea modificar Contraseña? S/N

Pide contraseña.

Confirma la modificación S/N:

4-Salir

2-Posteos

Ingresar posteo

Usuario nro:

Valida usuario

Pide posteo:

3-Listados

1-Amigos

Pide Usuario nro:

Muestra amigos, nombre y apellido.

2-Listar posteo

Pide Usuario nro:

Muestra comentarios y fecha.

4-Salir

1. Usuario: nombre, apellido, cedula identidad, correo electrónico y usuario y contraseña
2. Los usuarios pueden ser amigos de otros usuarios (se representa con un grafo)
3. Registrar posteos de su propia página. El posteo está identificado por un usuario. Tendrá texto y usuario.
4. Dado un usuario devolver la lista de amigos y posteos. La lista de amigos se desprende del grafo. Para la lista de posteos se ingresará un usuario y se listarán todos los posteos asociados a él.
5. Registrar cuantos usuarios hay en la red. Llevar un contador de usuarios o calcularlo cada vez que se ingresa uno nuevo ¿Deberemos guardarlo en alguna estructura?

TAD:

nombre=colección(carácter)

apellido=colección(carácter)

ci= entero

correoelectronico=colección(carácter)

user=colección(carácter)

contraseña=colección(carácter)

dia= entero

mes= entero

año= entero

fecha=diaxmesxaño

usuario= nombrexapellidoxcedulaidentidadxcorreoelectronicoxusuarioxcontraseña

usuarios=colección(usuario)

amigos=colección(usuario)

posteo=usuario x fechaxtexto

posteos=colección(posteo) (lista de adyacencia para cada usuario almacenar sus posteos)

amistad=personaxusuario

amistades=colección(amistad)

red\_social= UD (usuariosxamigosxposteos)

**TAD RED\_SOCIAL**

//crea una red\_social

**crear: Ø -> red\_social**

//añade un usuario

//PRE: NO debe existir el usuario previamente

//PRE: La red social NO puede estar llena

**add\_usuario: red\_social, usuario -> red\_social**

//Dado 2 usuarios añade una relación entre ellos \*

**amistad: red\_social, usuario, usuario -> red\_social**

//Devuelve una red\_social sin el usuario y las relaciones de amistad del usuario con otros

**borrar\_usuario: red\_social, usuario -> red\_social**

//Devuelve una red\_social sin la amistad indicada.

//Si la amistad ∄ devuelve la red\_social inicial

**borra\_amistad: red\_social, usuario, usuario -> red\_social**

//comprueba si una red\_social tiene usuarios

**es\_vacia: red\_social -> Bool**

//comprueba si una red\_social esta llena

**esta\_llena: red\_social -> Bool**

//comprueba si un usuario pertenece a una red\_social

**pertenece: red\_social, usuario -> Bool**

//comprueba si 2 usuarios son amigos

**amigos: red\_social, usuarios, usuario -> Bool**

//Nota: para la implementación debemos chequear que ambos usuarios existan

**DICCIONARIO PARA USUARIO**

| Cabezales de los operadores | Precondición | Post condición |
| --- | --- | --- |
| Make:Ø->DiccionarioUsu | ----------------- | Crea el diccionario vacío |
| Member:DiccionarioUsuxK->Boolean | ---------------- | Devuelve verdadero si la clave existe |
| Insert:DiccionarioUsuxT->DiccionarioUsu | El elemento a insertar NO está en el diccionario | Agrega un elemento en el diccionario |
| Find:DiccionarioUsuxK->T | El elemento es miembro del diccionario | Agrega un elemento en el diccionario |
| Modify:DiccionarioUsuxT-DiccionarioUsu | El elemento debe estar en el diccionario | Devuelve el elemento de la clave K |
| Delete:DiccionarioUsuxK->DiccionarioUsu | El elemento es miembro del diccionario | Elimina el elemento que corresponde a esa clave K |