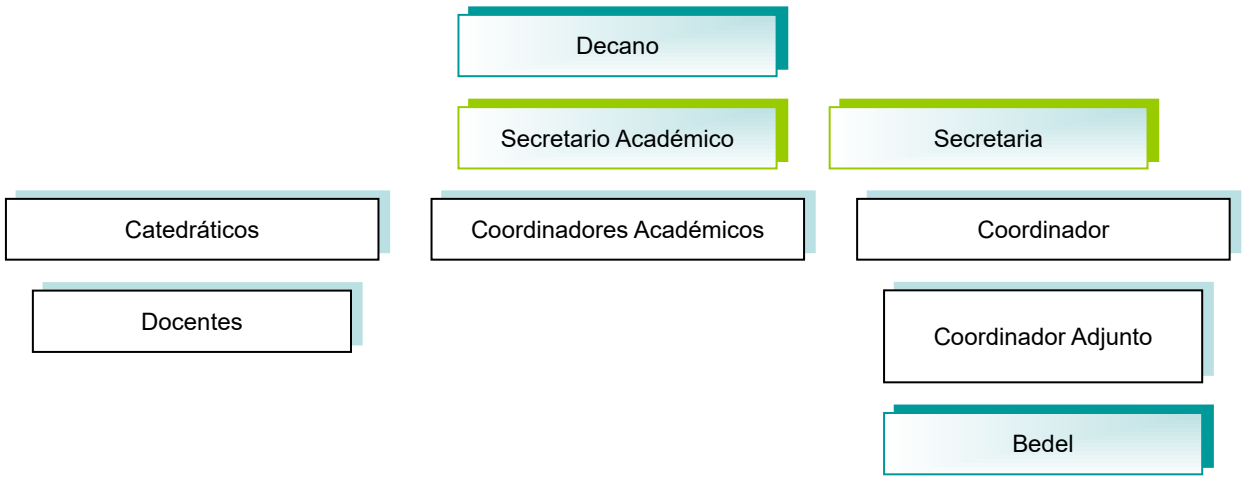
Trabajo Obligatorio de
Estructuras de Datos y Algoritmos
Organigrama

Se desea implementar un sistema que permita representar el organigrama de una empresa. Por el momento, sólo se consideran relaciones funcionales. Como características consideramos:

1) Los cargos de la empresa se identifican por su nombre, no pudiendo existir dos cargos con el mismo nombre.
2) Se deben ignorar mayúsculas y minúsculas, tanto para nombres de cargos como para nombres de personas.
3) Las personas se identifican por su cédula de identidad.
4) Siempre existe un único cargo que representa la jerarquía máxima de la empresa.
5) Cada cargo puede tener 0, 1 ó más subcargos dependientes de él.
6) Un cargo depende de un sólo cargo.
7) Cada cargo esta ocupado por 0, 1 o más personas pertenecientes a la empresa.
8) En un momento dado, una persona puede ocupar a lo sumo un cargo en la empresa, aunque éste puede cambiar a lo largo del tiempo.

Ejemplo:



En el ejemplo, el organigrama representa posibles cargos en una facultad. El cargo que representa la jerarquía máxima es “Decano”. En este ejemplo no figuran los empleados asignados a cada cargo.

Tipos de datos a manejar:

Cadena	`typedef char* Cadena;`

TipoRet	`enum tipo_retorno{`
	`  OK, ERROR, NO_IMPLEMENTADA`
	`};`
	
	`typedef enum tipo_retorno TipoRet;`

Persona	`struct tipo_persona{`
	`  Cadena ci;`
	`  Cadena nom;`
	`};`
	`typedef tipo_persona* Persona;`

Empresa	`struct tipo_empresa{`
	`  /*aquí deben figurar los campos`
	`  que usted considere necesarios`
	`  para manipular la empresa*/`
	`};`
	`typedef tipo_empresa* Empresa;`

Pueden (deben) definirse tipos de datos (estructuras de datos) auxiliares.

El sistema debe permitir realizar las siguientes operaciones:

OPERACIONES RELATIVAS A LOS CARGOS:

1) Crear el organigrama asignando un nombre al primer cargo "cabeza" de la jerarquía.

   `TipoRet CrearOrg(Empresa &e, Cadena cargo);`

   Inicializa la empresa y crea el primer **`cargo`** de la empresa. Originalmente la misma debería estar vacía, en otro caso la operación quedará sin efecto.

   ```
   Ejemplo:
         CrearOrg(e, "Decano");
         ListarJerarquia(e);

         Salida:
               Decano
   ```

Retornos posibles:	
OK	• Si se pudo crear el primer cargo con éxito
ERROR	• Si la empresa ya contaba con cargos creados anteriormente.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

2) Eliminar el organigrama, elimina toda la estructura del organigrama, liberando la memoria asignada.

`TipoRet EliminarOrg(Empresa &e);`

```
Ejemplo:
      EliminarOrg(e);
      ListarJerarquia(e);

      Salida:
            ERROR: empresa vacía.
```

Retornos posibles:	
OK	• Si se pudo eliminar correctamente la empresa.
ERROR	• Si la empresa se encuentra vacía.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

3) Insertar un nuevo cargo como dependiente de otro ya existente. El nuevo cargo no debe existir en el sistema.

`TipoRet NuevoCargo(Empresa &e, Cadena cargoPadre, Cadena nuevoCargo);`

Crea un nuevo cargo en la empresa si **`cargoPadre`** ya existe en la misma y **`nuevoCargo`** no. En otro caso la operación quedará sin efecto.

```
Ejemplo:
      NuevoCargo(e,"Decano", "Secretario academico");
      NuevoCargo(e,"Decano", "Secretaria");
      NuevoCargo(e,"Secretario academico", "Catedratico");
      ListarJerarquia(e);

      Salida:
            Decano
                  Secretario academico
                        Catedratico
                  Secretaria
```

Retornos posibles:	
OK	• Si se pudo crear el **`nuevoCargo`** con éxito
ERROR	• Si **`cargoPadre`** no existe. En particular cuando la empresa está vacía. • Si **`nuevoCargo`** ya existe.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

4) Eliminar un cargo, junto con sus subcargos y personas asociadas.

`TipoRet EliminarCargo(Empresa &e, Cadena cargo);`

Elimina un cargo en la empresa si **`cargo`** ya existe en la misma. En otro caso la operación quedará sin efecto. Si el cargo a eliminar posee subcargos, éstos deberán ser eliminados también, así como las personas asociadas a cada uno de los cargos suprimidos (incluyendo las personas asociadas a **`cargo`**).

```
Ejemplo:
      EliminarCargo(e, "Secretario academico");
      ListarJerarquia(e);
            Salida:
            Decano
                  Secretaria

      EliminarCargo(e, "Decano");

      Salida:
            ERROR: La empresa no debe estar vacia
```

Retornos posibles:	
OK	• Si se pudo eliminar el **cargo** con éxito
ERROR	• Si **cargo** no existe. En particular cuando la empresa está vacía.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

5) Listar todos los cargos ordenados alfabéticamente.

`TipoRet ListarCargosAlf(Empresa e);`

Lista todos los cargos de la empresa ordenados alfabéticamente por nombre del cargo.

```
Ejemplo:
      NuevoCargo(e,"Decano", "Secretario academico");
      NuevoCargo(e,"Decano", "Secretaria");
      NuevoCargo(e,"Secretario academico", "Catedratico");
      NuevoCargo(e,"Secretario academico", "Coordinador academico");
      NuevoCargo(e,"Secretario academico", "Coordinador");
      NuevoCargo(e,"Catedratico", "Docente");
      NuevoCargo(e,"Coordinador", "Coordinador adjunto");
      NuevoCargo(e,"Coordinador adjunto", "Bedel");
      ListarCargosAlf(e);

Salida:
      Bedel
      Catedratico
      Coordinador
      Coordinador academico
      Coordinador adjunto
      Decano
      Docente
      Secretaria
      Secretario academico
```

Retornos posibles:	
OK	• Si existe por lo menos un cargo creado, la empresa no está vacía
ERROR	• Si se ejecuta cuando la empresa está vacía.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

6) Listar todos los cargos de la empresa en orden jerárquico.

`TipoRet ListarJerarquia(Empresa e);`

Lista todos los cargos de la empresa ordenados por nivel jerárquico e indentados según muestra el siguiente ejemplo.

```
Ejemplo:
      ListarJerarquia(e);

Salida:
      Decano -
        Secretaria -
        Secretario academico -
          Coordinador -
            Coordinador adjunto -
              Bedel -
          Coordinador academico -
          Catedratico -
            Docente -
```

Retornos posibles:	
OK	• Si existe por lo menos un cargo creado (la empresa no está vacía).
ERROR	• Si se ejecuta cuando la empresa está vacía.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

OPERACIONES RELATIVAS A LAS PERSONAS:

7) Asignar una persona a un cargo, si este existe.

`TipoRet AsignarPersona(Empresa &e, Cadena cargo, Cadena nom, Cadena ci);`

Asigna una persona de nombre **`nom`** y cédula de identidad **`ci`** al cargo **`cargo`** siempre que el cargo exista en la empresa y esa persona no este asignada a ese u otro cargo, en caso contrario la operación quedará sin efecto.

```
Ejemplo:
      AsignarPersona(e, "Decano", "Empleado4", "4444444");
      AsignarPersona(e, "Decano", "Empleado5", "5555555");
      AsignarPersona(e, "Decano", "Empleado6", "6666666");
      ListarPersonas(e, "Decano");

Salida:
      Listado de personas asignadas a Decano
      ---------------------------------------
      4444444 - Empleado4
      5555555 - Empleado5
      6666666 - Empleado6
```

Retornos posibles:	
OK	• Si se pudo asignar la persona al cargo **`cargo`** con éxito
ERROR	• Si **`cargo`** no existe. En particular cuando la empresa está vacía. • Si la persona con cédula **`ci`** ya existe en ese u otro cargo.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

8) Eliminar una persona de un cargo.

`TipoRet EliminarPersona(Empresa &e, Cadena ci);`

Elimina una persona de cédula **`ci`** de la empresa siempre y cuando la misma exista, en caso contrario la operación quedará sin efecto.

```
Ejemplo:
      EliminarPersona(e, "5555555");
      ListarPersonas(e, "Decano");

Salida:
      Listado de personas asignadas a Decano
      ---------------------------------------
      4444444 - Empleado4
      6666666 - Empleado6
```

Retornos posibles:	
OK	• Si se pudo eliminar la persona de cédula **`ci`** con éxito
ERROR	• Si la persona de cédula **`ci`** no existe. En particular cuando la empresa está vacía.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

9) Reasignar una persona a un nuevo cargo.

`TipoRet ReasignarPersona(Empresa &e, Cadena cargo, Cadena ci);`

Reasigna una persona de la empresa de cédula **`ci`** al nuevo cargo de nombre **`cargo`** siempre que el cargo exista en la empresa y esa persona no este ya asignada a dicho cargo. En caso contrario la operación quedará sin efecto.

```
Ejemplo:
      ReasignarPersona(e, "Coordinador", "4444444");
      ListarPersonas(e, "Coordinador");
      ListarPersonas(e, "Decano");
Salida:
      Listado de personas asignadas a Coordinador
      -------------------------------------------
      4444444 - Empleado4

      Listado de personas asignadas a Decano
      --------------------------------------
      6666666 - Empleado6
```

Retornos posibles:	
OK	• Si se pudo reasignar la persona de cédula **`ci`** al nuevo cargo **`cargo`**
ERROR	• Si **`cargo`** no existe. En particular cuando la empresa está vacía. • Si la persona de cédula **`ci`** no existe • Si la persona ya estaba asignada a ese mismo cargo.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

10) Dado un cargo listar las personas asignadas al mismo ordenadas por fecha de alta a la empresa.

```
TipoRet ListarPersonas(Empresa e, Cadena cargo);
```

Lista todas las personas asignadas al cargo de nombre **cargo**.

Retornos posibles:	
OK	• Si existe el cargo **cargo** buscado.
ERROR	• Si **cargo** no existe. En particular cuando la empresa está vacía.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

11) Dado un cargo listar los cargos que lo anteceden

```
TipoRet ListarSuperCargos (Empresa e, Cadena cargo);
```

Lista todas los cargos que anteceden, en la jerarquía, al cargo de nombre **cargo**.

```
Ejemplo:
      ListarSuperCargos(e, "Bedel");

Salida:
      Decano -
      Secretario academico -
      Coordinador -
      Coordinador adjunto -
      Bedel -
```

Retornos posibles:	
OK	• Si existe el cargo **cargo** buscado.
ERROR	• Si **cargo** no existe. En particular cuando la empresa está vacía.
NO_IMPLEMENTADA	• Cuando aún no se implementó. Es el tipo de retorno por defecto.

CATEGORÍA DE OPERACIONES

TIPO 1	Operaciones **imprescindibles** para que el trabajo obligatorio sea corregido.
TIPO 2	Operaciones importantes. Estas serán probadas independientemente, siempre que estén correctamente implementadas las operaciones de TIPO 1.

TIPO 1	TIPO 2
CrearOrg (Control Intermedio)	**EliminarCargo**
NuevoCargo	**ListarJerarquia**
ListarCargosAlf	**EliminarPersona (Control Intermedio)**
AsignarPersona (Control Intermedio)	**ReasignarPersona**
ListarPersonas (Control Intermedio)	**ListarSuperCargos**
EliminarOrg(Control Intermedio)	

A FIN DE CONSTRUIR LAS IMPLEMENTACIONES ANTERIORES SE PIDE:

- Definición y desarrollo de las estructuras de datos involucradas, siguiendo la metodología basada en módulos vista en el curso, y usando C/C++ como lenguaje de programación.
- Implementación de los módulos. El código de todo el sistema debe estar desarrollado y comentado de acuerdo a las prácticas presentadas en el curso. Incluir las pre- y pos-condiciones de las operaciones.
- Se dispone de un main y un pequeño esqueleto, las pruebas para las correcciones se realizaran utilizando este main.
- Se debe respetar estrictamente las salidas

PLAZOS Y FORMA DE ENTREGA:

- La tarea deberá realizarse en **grupos de tres estudiantes** cada uno. Los grupos deben inscribirse llenando un formulario web hasta el 10 de octubre inclusive, los estudiantes que se anoten de forma individual se les asignará un grupo de forma aleatoria.
- **Entregas:**
    - **Control Intermedio:**
        - Se debe coordinar un **monitoreo antes del 25 de octubre**, para recibir una retroalimentación sobre las operaciones exigidas para esta instancia.
        - Si trabajos son completamente insuficientes en esta instancia puede implicar la perdida del laboratorio con la consecuente pérdida del curso.
    - **Entrega Final:**
        - La fecha para la **entrega final** será el **20 de Noviembre de 2024** hora 23:59.
        - Deben entregar vía Campus Virtual, siguiendo estrictamente el formato especificado.
        - Deben coordinar una defensa con todos los integrantes del grupo.
- A quienes realicen las operaciones Tipo 2 se les podrán asignar hasta 10 puntos extra en el puntaje del curso.