Trabajo práctico Final

1. Motivación

El trabajo involucra el desarrollo de un programa para el manejo de una *Agenda de Contactos*. El programa puede correr por línea de comandos y dispone de una interfaz de usuario que permite manipular los datos almacenados. La agenda debe almacenar datos para los siguientes atributos: nombre, apellido, edad, y número de teléfono. Suponemos que cada contacto se identifica unívocamente por su nombre y apellido, es decir, no habrá dos contactos con mismo nombre y apellido.

2. Estructura contacto:

Un contacto se define con la siguiente estructura:

```
struct _Contacto {
  char *nombre;
  char *apellido;
  unsigned edad;
  char *telefono;
};
typedef struct _Contacto *Contacto;
```

3. Acciones

A continuación se describen las acciones a ser provistas por el programa:

1. **Buscar**: pide al usuario ingresar un nombre y un apellido, busca en la agenda el contacto correspondiente e imprime por pantalla sus datos.
2. **Agregar**: pide al usuario ingresar los datos de un nuevo contacto y lo agrega a la agenda.
3. **Eliminar**: pide al usuario ingresar un nombre y un apellido y elimina de la agenda el contacto correspondiente.
4. **Editar**: pide al usuario un nombre y un apellido, si el contacto se encuentra en la agenda entonces pide los demás datos y los reemplaza.
5. **Cargar:** pide al usuario ingesar la ruta de un archivo de entrada en formato `CSV` con datos de contactos y carga todos ellos en la agenda. El volumen de datos a cargar puede ser variable. El archivo debe contener las siguientes columnas: `nombre,apellido,edad,telefono`. A continuación se muestra un ejemplo de un archivo que contiene datos de 2 contactos:

   ```
   nombre,apellido,edad,telefono
   Sabrina,Perez Garcia,18,3418796441
   Juan Manuel,Gonzalez,42,3414521201
   ```

6. **Guardar:** pide al usuario ingresar la ruta de un archivo de salida en formato `CSV` y escribe en él los datos de todos los contactos de la agenda, respetando el formato detallado en el punto anterior.

7. **Deshacer/Rehacer:** deshace o rehace respectivamente el cambio más reciente producido en la agenda (los cambios son realizados por las acciones: agregar, eliminar, y editar). Debe ser posible deshacer los últimos N cambios realizados, de a uno a la vez, donde N es una constante definida en el programa por medio una directiva de preprocesador (análogo para rehacer).

8. **AND/OR:** pide al usuario ingresar un valor para cada atributo (puede ingresar un valor vacío), busca en la agenda el/los contactos cuyos atributos coincidan con todos los valores buscados (AND) o con alguno de los valores buscados (OR), e imprime por pantalla su/sus datos. El usuario debe ingresar un valor no vacío para al menos uno de los atributos. Por ejemplo si el usuario ingresa nombre Carlos, apellido vacío, edad 75, y teléfono vacío, entonces AND imprime todos los contactos de la agenda que se llamen Carlos y tengan 75 años, mientras que OR imprime todos los contactos que se llamen Carlos o que tengan 75 años.

9. **Guardar ordenado**: similar a guardar, salvo que pide además al usuario ingresar un nombre de atributo (nombre, apellido, edad, o teléfono) y escribe los contactos ordenados según dicho atributo.

10. **Buscar por suma de edades:** pide al usuario ingresar un número natural, busca un subconjunto de contactos de la agenda cuyas edades sumen en total el número dado, e imprime por pantalla el/los contactos del subconjunto.

11. **Salir:** cierra el programa y libera los recursos solicitados.

Nota: puede suponer que los nombres y apellidos contienen únicamente caracteres ASCII, es decir, sin acentos, letras ñ, etc.

4. Interfaz de usuario

El programa dispone de una interfaz de usuario que funciona por entrada/salida estándar. Al inicio, muestra un menú con las posibles acciones. Inmediatamente, solicita al usuario que ingrese el número de la acción que desean invocar. En caso de necesitar parámetros adicionales, los solicita y realiza la acción correspondiente. A continuación se muestra un ejemplo de ejecución:

```
Menu de acciones:
1. Buscar
2. Agregar
3. Eliminar
4. Editar
5. Cargar
6. Guardar
7. Deshacer
8. Rehacer
9. AND
10. OR
11. Guardar ordenado
12. Buscar por suma de edades
13. Salir

Seleccione una accion:
>5
```

```
Ingrese ruta de entrada:
>entrada/agenda.csv

Seleccione una accion:
>2
Ingrese nombre:
>Carlos
Ingrese apellido:
>Rodriguez
Ingrese edad:
>75
Ingrese telefono:
>3419910412

Seleccione una accion:
>12
Ingrese un natural:
>117
{Juan Manuel,Gonzalez,42,3414521201}
{Carlos,Rodriguez,75,3419910412}

Seleccione una accion:
>11
Ingrese ruta de salida:
>salida/agenda.csv
Ingrese nombre de atributo:
>apellido

Seleccione una accion:
>13
```

En este ejemplo la primera acción realizada es cargar una agenda desde el archivo `entrada/agenda.csv`, vamos a suponer que este archivo es el que se presentó al describir la acción cargar. Luego, se ingresa un nuevo contacto a la agenda de nombre Carlos, apellido Rodriguez, edad 75, y número de teléfono 3419910412. Luego, se busca un subconjunto de contactos cuyas edades sumen 117 y se imprime el conjunto resultante. Luego, se imprime la agenda en el archivo `salida/agenda.csv` con los contactos ordenados por apellido, produciendo el siguiente contenido:

```
nombre,apellido,edad,telefono
Juan Manuel,Gonzalez,42,3414521201
Sabrina,Perez Garcia,18,3418796441
Carlos,Rodriguez,75,3419910412
```

Por último, se cierra el programa.

5. Consigna

1. Diseñe e implemente un programa que cumpla con las funcionalidades detalladas. Considere los siguientes puntos:

   - Los requerimientos no mencionado deben ser resueltos a criterio personal.
   - Respete las convenciones de código y proyecto elaboradas por la cátedra y disponibles en comunidades.
   - Consultas: **únicamente** por mensaje privado de zulip. Usar como destinatario `edya1`.

2. Incluya en la entrega los casos de prueba usados para probar el programa.

3. Confeccione un informe detallando brevemente **cada uno** de los siguientes puntos.

   a) Módulos en los que está dividido el programa.

   b) Estructuras de datos utilizadas, justificando adecuadamente la elección.

   c) Algoritmos que implementan las acciones: deshacer/rehacer, guardar ordenado, y buscar por suma de edades.

   d) Dificultades encontradas y cómo fueron resueltas.

   e) Cómo se compila e invoca el programa.

   f) Bibliografía consultada.