Parcialito Proyecto 2 - Tema A

1. Modificar el TAD `list` tal que éste provea una operación nueva cuya signatura es:

   ```
   int list_position(list_t list, index_t index)
   ```

   Esta operación devuelve la posición en la lista en donde se encuentra el primer par cuyo índice es `index`. Si en la lista no hay ningún par cuyo índice es igual al dado, se debe retornar el valor `-1`.

   Para la implementación de este método **no** se deben hacer llamadas a otros métodos públicos del TAD lista. Es decir, no se puede llamar a `list_copy`, `list_remove`, `list_search`, etc. (excepto para escribir aserciones de PRE y POST). Por supuesto que se pueden usar todos los métodos públicos de los otros TADs, como el index, el data, el pair.

   Las PRE y POST de este método nuevo son las siguientes:

   pre la lista `list` y el índice `index` dados son válidos. El índice puede **o no** estar en la lista.

   post el resultado es exactamente la posición del primer par cuyo índice es igual a `index`, o `-1` si tal par no existe. Las posiciones se numeran como en C: la primer posición es 0, y la última N-1 (asumiendo que la lista tiene N elementos). Por ende, vale que:
   `(list_search(list, index)` $\neq$ `NULL` $\Rightarrow$ $0 \leq$ `result` $<$ `list_length(list))` $\wedge$
   `(list_search(list, index)` $=$ `NULL` $\Rightarrow$ `result = -1)`

   Ejemplo supongamos una lista con los siguientes 4 pares:

   ```
   [("perro", "cuadrupedo que ladra"), ("gato", "cuadrupedo que maulla"),
    ("loro", "alumno que habla en clase"), ("perro", "el mejor amigo del hombre")]
   ```

   El nuevo método `list_position` debería retornar los siguientes resultados:

   - 0 para `"perro"`
   - 1 para `"gato"`
   - 2 para `"loro"`
   - -1 para `"tortuga"` (o cualquier otra palabra que no esté en la lista)

   OJO Qué retorna `list_position` para la lista vacía?

2. Crear un archivo `parcialito.c` que provea una función `main` que haga lo siguiente:

   - Crear una lista vacía y agregarle los siguientes elementos:
     - ("caza", "Accion de cazar")
     - ("casa", "Edificio para habitar")
     - ("cielo", "Esfera aparente azul que rodea la Tierra")
     - ("caza", "Conjunto de animales no domesticados")
     - ("extraordinario", "Fuera del orden o regla natural")

     Ayuda: Para crear índices y datos usando cadenas de texto estáticas como las dadas arriba (es decir, strings que no usan memoria dinámica, por ende no hay que liberarlos), pueden hacer lo siguiente:

     ```
     list = list_append(list, index_from_string("perro"),
                        data_from_string("animal que ladra"));
     /* no hay que liberar ni el index ni el data, ni los strings estaticos */
     ```

- Llamar al método implementado en el punto 1 (`list_position`) tal que se muestre por pantalla las posiciones de los índices:
  - "extraordinario"
  - "caza"
  - "zaraza"
  - "cielo"

El mensaje con el resultado tiene que estar terminado en `"\n"` y tiene que contener la palabra `position`, la clave usada y el número de la posición. Ejemplo de mensajes válidos:

```
The position for loro is: 2
```

```
The position for zaraza is: -1
```

3. Recordar

   - Se debe resolver el ejercicio en una hora (o menos).
   - Se debe compilar usando `gcc` desde la consola pasando todos los flags usados en el proyecto (no hace falta hacer un Makefile).
   - Comentar e indentar el código apropiadamente, siguiendo el estilo de código ya provisto por la cátedra en los archivos dados (indentar con 4 espacios, no pasarse de las 80 columnas, inicializar todas las variables, etc).
   - Todo el código tiene que usar la librería estándar de C, y no se puede usar extensiones GNU de la misma.
   - El programa resultante **no** debe tener *memory leaks* **ni** accesos (read o write) inválidos a la memoria.