CS2023 - Aula de Ejercicios Nº 3

Heider Sanchez
ACL: Juan Diego Castro
Semestre 2024-1

Se sugiere que cada estudiante trate de resolver los ejercicios de forma **individual** y luego los discuta en grupo.

Si el estudiante obtiene más de 20 puntos, los puntos adicionales pueden pasarse a otra continua.

Ejercicios

1. (8 pts) Diseña una estructura de datos que utilice una lista doblemente enlazada para almacenar strings y un contador por cada string. La estructura debe tener la capacidad de retornar el elemento con el contador máximo y mínimo.

   - Deben implementar su propio struct para nodos doblemente enlazados

   Implementa la clase AllOne:

```
class AllOne {
public:
    AllOne() {}
    void inc(string key) {}
    void dec(string key) {}
    string getMaxKey() {}
    string getMinKey() {}
};
```

   - AllOne() Inicializa la estructura
   - inc(String key) Incrementa el contador del string 'key' en 1. Si no existe, lo inserta en la lista con contador 1
   - dec(String key) Disminuye el contador del string 'key' en 1. Si el contador se vuelve 0 luego de disminuirlo, eliminar el string de la lista. Puedes asumir que el string existirá en la lista antes de disminuir su contador.
   - getMaxKey() Retorna el elemento con contador máximo, si la lista está vacía, retornar string vacío.
   - getMinKey() Retorna el elemento con contador mínimo, si la lista está vacía, retornar string vacío.

   Ejemplo:

   Inputs:

```
["AllOne", "inc", "inc", "getMaxKey", "getMinKey", "inc", "getMaxKey", "getMinKey"]
[[], ["hello"], ["hello"], [], [], ["leet"], [], []]
```

   Output:

```
[null, null, null, "hello", "hello", null, "hello", "leet"]
```

   Restricciones:

   - $1 \leq key.length \leq 10$
   - Key sólo contiene letras del alfabeto inglés en minúsculas

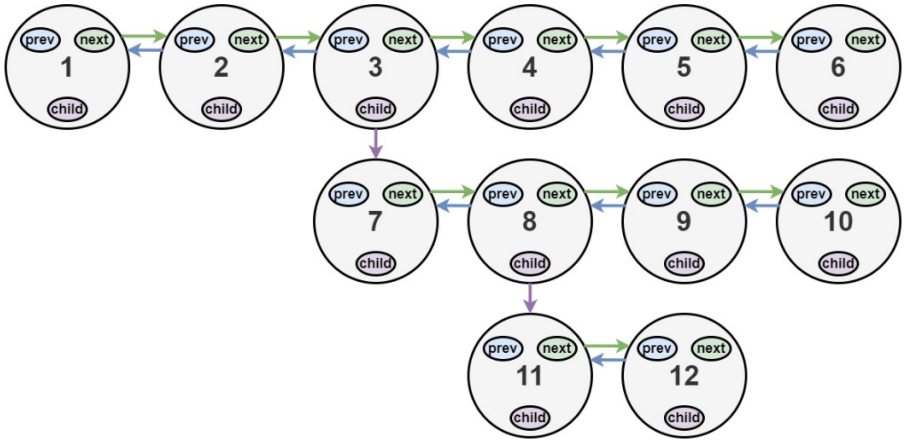
- Está garantizado que para cada llamada a dec, el string 'key' existe en la lista

2. (8 pts) Se le proporciona una lista doblemente enlazada, que contiene nodos que tienen un puntero siguiente, un puntero anterior y un puntero secundario adicional `child`. Este puntero secundario puede o no apuntar a una lista doblemente enlazada separada, que también contiene estos nodos especiales. Estas listas secundarias pueden tener uno o más hijos propios, y así sucesivamente, para producir una estructura de datos multinivel como se muestra en el siguiente ejemplo.

   Dado el *head* del primer nivel de la lista, aplane la lista para que todos los nodos aparezcan en una lista doblemente enlazada de un solo nivel. Sea `curr` un nodo con una lista de hijos. Los nodos en la lista secundaria deben aparecer después de `curr` y antes de `curr.next` en la lista aplanada.

   Devuelve el *head* de la lista aplanada. Los nodos de la lista deben tener todos sus punteros secundarios establecidos en nulo.

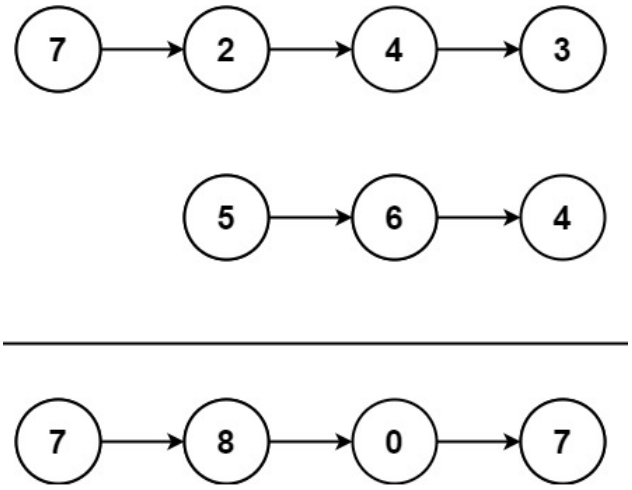
   Input:



   Output: [1,2,3,7,8,11,12,9,10,4,5,6]

   Explicación: Se muestra la lista enlazada multinivel en la entrada. Después de aplanar la lista enlazada multinivel se convierte en:

3. (8 pts) Recibes como entrada dos listas simplemente enlazadas no vacías que representan dos números no negativos. El dígito más significativo es el primer elemento de la lista. Retorna la suma de ambos números en una nueva lista. Puedes asumir que nunca habrá un 0 como primer elemento, excepto por la lista que representa el número 0.



- Ejemplo 1:
  $Input : l1 = [7, 2, 4, 3], l2 = [5, 6, 4]$
  $Output : [7, 8, 0, 7]$
- Ejemplo 2:
  $Input : l1 = [2, 4, 3], l2 = [5, 6, 4]$
  $Output : [8, 0, 7]$
- Example 3:
  $Input : l1 = [0], l2 = [0]$
  $Output : [0]$