Abel Anaya C.I. 25205438

Se implementa una lista doblemente enlazada con apuntadores a siguiente y anterior al nodo y un entero (marca). La lista tiene 3 apuntadores al primer nodo, el medio y el último y un entero (espadas) que es el número de nodos en la lista. Las estructuras que se usaron fueron lista(), insertar(entero), tomar(), reordenar(), mostrar().

Lista(): es el constructor, inicializa los apuntadores en NULL y el tamaño de la lista en 0. Es de orden O(1) ya que solo se realizan asignaciones.

Void insertar(entero): inserta un nodo con un entero (marca) en la lista y se suma a la cantidad de nodos en la lista (espadas). Crea un nuevo nodo con sus apuntadores anterior y último apuntando a NULL, luego evalúa los diferentes casos que pasan cuando se inserte un nuevo nodo y realiza la inserción dependiendo del caso:

- Si es el primer nodo de la lista hace que los apuntadores primero, medio y último apunten hacia el nuevo nodo.
- Si es el segundo nodo o un nuevo nodo que se le suma a una lista par, hace que el último nodo apunte al nuevo nodo, el anterior del nuevo apunte al último nodo de la lista y el apuntador ultimo apunte al nuevo nodo.
- Si es un nuevo nodo que se le suma a una lista impar, hace lo mismo que el paso anterior y además al apuntador medio se cambia para que apunte a su siguiente nodo.

Después de ingresar el nodo se le suma 1 a la cantidad de espadas. Como se realizan únicamente asignaciones y una declaración es de orden O(1).

Void tomar(): Elimina un nodo de la lista y se le resta a las espadas. Crea un apuntador auxiliar apuntando al mismo nodo que el apuntador último, luego evalúa los diferentes casos que pasan cuando se elimine un nodo y elimina dependiendo del caso:

- Si solo hay un nodo hace que los apuntadores primero, medio y último apunten a NULL y borra el nodo apuntado por auxiliar.
- Si hay dos nodos o la lista es impar hace que el apuntador último apunte al anterior del último nodo, hace que el próximo de ese nodo apunte a NULL y borra el nodo apuntado por auxiliar.
- Si la lista es par, hace lo mismo que el paso anterior y además al apuntador medio se le asigna el anterior nodo.

Después de eliminar el nodo se le resta 1 a la cantidad de espadas. Como se realizan únicamente asignaciones y una declaración es de orden O(1).

Void reordenar(): Cambia el orden de la lista tal que la “primera mitad” seria la “última mitad”. Crea un apuntador auxiliar apuntando al mismo nodo que el apuntador último, luego evalúa los diferentes casos que pasan cuando se reordena la lista y reordena dependiendo del caso:

- Si la lista tiene una sola espada no hace nada.
- Si la lista tiene una cantidad par de espadas, el apuntador ultimo ahora apuntado al mismo nodo que el apuntador medio, el próximo del auxiliar apunta al mismo nodo que el apuntador primero, el anterior del primero apunta al mismo nodo que el apuntador auxiliar, el apuntador primero apunta al próximo del nodo medio, el apuntador medio apunta al mismo nodo que el apuntador auxiliar, el próximo del nodo ultimo apunta a NULL y el anterior del nodo primero apunta a NULL.
- Si la lista tiene una cantidad impar de espadas, hace lo mismo que el paso anterior con la diferencia de que el apuntador medio apunta al anterior del nodo auxiliar.

Como se realizan únicamente asignaciones y una declaración es de orden O(1).

Void mostrar(): muestra la cantidad de nodos (espadas) y todos los datos (marca) de la lista. Crea un apuntador auxiliar para recorrer la lista, es de orden O(n) ya que depende de cuantos elementos tenga la lista.

En int main() se obtiene un entero n de hasta 1000000 que será el numero de instrucciones y se lee un string por cada instrucción que alterna; si es “insertar” lee un entero y llama a la función insertar, si es “tomar” llama a la función tomar y si es “reordenar” llama a la función reordenar; al finalizar la cantidad de instrucciones llama a la función mostrar. El algoritmo finaliza con orden O(n).