Báo cáo bài tập về nhà tuần 4

1 Linkedlist

Given the following Linkedlist definition:

```
struct NODE{ int key;
        NODE* p_next;
};
```

```
struct List{
        NODE* p_head;
        NODE* p_tail;
};
```

Complete the following functions to fulfill the given requirements:

1. Initialize a NODE from a given integer:

    - NODE* createNode(int data)

- Tạo 1 node mới, gán key = data, next = nullptr.

2. Initialize a List from a given NODE:

    - List* createList(NODE* p_node)

- Tạo 1 list mới, gán head = tail = p_node.

3. Insert an integer to the head of a given List:

    - bool addHead(List* &L, int data)

- Th1: list không tồn tại, trả về false
- Th2: list tồn tại
  + Tạo một node p mới chứa dữ liệu cần thêm
  + Nối p vào list
  + Cập nhật head, nếu tail là nullptr, tail = head

4. Insert an integer to the tail of a given List:

    - bool addTail(List* &L, int data)

- Th1: list rỗng
  + Tạo newNode chứa dữ liệu cần thêm
  + cập nhật head và tail bằng newNode
- Th2: list khác rỗng
  + nối tail với newNode

+ cập nhật tail

5. Remove the first NODE of a given List:

- bool removeHead(List* &L)

- Th1: nếu list rỗng hoặc không tồn tại thì trả về false
- Th2: list khác rỗng
  + tạo node temp chứa head
  + cập nhật head = head->next
  + xóa temp
  + nếu sau khi xóa thì list rỗng thì cần cập nhật lại tail = nullptr

6. Remove the last NODE of a given List:

- void removeTail(List* &L)

- Th1: nếu list rỗng hoặc không tồn tại thì trả về.
- Th2: nếu list chỉ có một phần tử
  + xóa tail và cập nhật head = tail = nullptr
- Th3:
  + tạo node p chứa head
  + dịch chuyển p đến node trước tail
  + gán node temp là node chứa tail
  + cập nhật tail và tail->next = nullptr
  + xóa temp

7. Remove all NODE from a given List:

- void removeAll(List* &L)

- Th1: nếu list rỗng hoặc không tồn tại thì trả về.
- Th2: list khác rỗng
  + tạo node p chứa head
  + dịch chuyển p đến khi p = nullptr, trong khi đó gán temp = p và xóa temp khi dịch chuyển p sang node tiếp theo
  + cập nhật tail và head = nullptr

8. Remove a Node Before with a given value List:

- void removeBefore(List* &L, int val)

- Th1: nếu list rỗng hoặc không tồn tại thì trả về.
- Th2: list khác rỗng
  + tạo node p chứa head
  + nếu node chứa dữ liệu cần tìm là head thì return
  + nếu node chứa dữ liệu cần tìm là node ngay sau head: xóa và cập nhật head, return

+ dịch chuyển p đến node phía trước node cần xóa hoặc đến node kề cuối của list
+ nếu p là node kề cuối của list thì return;
+ gán temp là node tiếp theo của p tức là node cần xóa
+ cập nhật p->next để bỏ qua node temp
+ xóa temp

9. Remove an integer after a value of a given List:

- void removeAfter(List* &L, int val)

- Th1: nếu list rỗng hoặc không tồn tại thì trả về.
- Th2: list khác rỗng
  + tạo node p chứa head
  + dịch chuyển p đến node chứa dữ liệu cần tìm hoặc đến node cuối của list
  + nếu p là node cuối của list thì return;
  + gán temp là node tiếp theo của p tức là node cần xóa
  + cập nhật p->next để bỏ qua node temp
  + xóa temp
  + nếu node đã xóa là tail thì cần cập nhật lại tail

10. Insert an integer at a position of a given List:

- bool addPos(List* &L, int data, int pos)

- Th1: nếu (list rỗng và pos khác 0) hoặc list không tồn tại thì trả về.
- Th2: pos = 0
  + tạo node temp chứa data
  + nối temp vào head
  + cập nhật head
  + nếu ban đầu list rỗng thì cập nhật cả tail
- Th3:
  + tạo node p chứa head
  + dịch chuyển p đến node phía trước vị trí cần chèn hoặc đến node cuối của list
  + nếu p là node cuối của list thì return false;
  + tạo node temp chứa data
  + nối temp với list
  + nếu temp là node cuối cùng của list thì cần cập nhật lại tail

11. Remove an integer at a position of a given List:

- void RemovePos(List* &L, int data, int pos)

- Th1: nếu list rỗng hoặc list không tồn tại thì trả về.
- Th2: pos = 0
  + tạo node temp chứa head

+ cập nhật head
+ xóa temp
+ nếu sau khi xóa, list rỗng thì cập nhật cả tail

- Th3:
  + tạo node p chứa head
  + dịch chuyển p đến node phía trước vị trí cần xóa hoặc đến node cuối của list
  + nếu p là node cuối của list thì return
  + tạo node temp chứa node cần xóa
  + cập nhật p->next để bỏ qua liên kết với temp
  + nếu temp là node cuối cùng của list thì cần cập nhật của tail

12. Insert an integer before a value of a given List:

- bool addBefore(List* &L, int data, int val)

- Th1: nếu list rỗng hoặc list không tồn tại thì trả về false
- Th2: nếu head là node chứa giá trị cần tìm
  + tạo node temp chứa data
  + nối temp với head
  + cập nhật head
  + return true
- Th3:
  + tạo node p chứa head
  + dịch chuyển p đến node phía trước node chứa dữ liệu cần tìm hoặc đến node cuối của list
  + nếu p là node cuối của list thì return
  + tạo node temp chứa data
  + nối temp vào list
  + return true

13. Insert an integer after a value of a given List:

- bool addAfter(List* &L, int data, int val)

- Th1: nếu list rỗng hoặc list không tồn tại thì trả về false
- Th2:
  + tạo node p chứa head
  + dịch chuyển p đến node chứa dữ liệu cần tìm hoặc đến node cuối của list
  + nếu p là node cuối của list thì return
  + tạo node temp chứa data
  + nối temp vào list
  + nếu temp là node cuối của list thì cần cập nhật tail

14. Print all elements of a given List:

- void printList(List* L)

- tạo node p chứa head
- dịch chuyển p đến khi p = nullptr trong khi in ra p->key và dấu space
- in ra dấu xuống dòng

15. Count the number of elements List:

- int countElements(List* L)

- Th1: nếu list rỗng hoặc list không tồn tại thì trả về 0
- Th2:
  + tạo node p chứa head
  + dịch chuyển p đến khi p = nullptr trong khi tăng giá trị cho count
  + return count

16. Create a new List by reverse a given List:

- List* reverseList(List* L)

- Th1: nếu list rỗng hoặc list không tồn tại thì trả về false
- Th2:
  + tạo node prev, curr, next với giá trị ban đầu lần lượt là nullptr, head và nullptr
  + cập nhật tail là head
  + dịch chuyển curr đến khi curr = nullptr, trong khi:
  Nối curr đến prev
  Cập nhật curr, prev, next để duyệt đến hết list
  + cập nhật head và return list

17. Remove all duplicates from a given List:

- void removeDuplicate(List* &L)

- Th1: nếu list rỗng hoặc list không tồn tại hoặc list chỉ có một phần tử thì trả về false
- Th2:
  + tạo node p chứa head
  + dịch chuyển p đến node cuối của list, trong khi:
  Nếu p và p->next có key bằng nhau: xóa p->next; nếu node đã xóa là node cuối của list thì cần cập nhật tail

18. Remove all key value from a given List:

- bool removeElement(List* &L, int key)

- Th1: nếu list rỗng hoặc list không tồn tại thì trả về false
- Th2: nếu head là nút cần xóa, xóa và cập nhật head, return true
- Th3:

+ tạo node p chứa head
+ dịch chuyển p đến node cuối của list hoặc đến trước nút cần xóa
+ nếu p là nút cuối của list thì return false
+ cập nhật p->next và xóa nút cần xóa, nếu nút đã xóa là nút cuối của list thì cập nhật tail

2 Doubly Linkedlist

Following is representation of a doubly linked list:

```
struct d_NODE{ int key;
       d_NODE*      pNext;
       d_NODE* pPrev;
};
```

```
struct d_List{ d_NODE*
       pHead;     d_NODE*
       pTail;
};
```

Implement functions to execute the operations from singly linkedlist section (section 1).

Thực hiện như Singly linkedlist, bổ sung thêm việc cập nhật node prev và chú ý các nút next có phải là nullptr để cập nhật tail và cập nhật node prev.

3 Git/Github