Lab 03: Danh sách liên kết, Stack & Queue

Thời gian dự kiến: 02 tuần

1 Danh sách liên kết (DSLK)

1.1 DSLK đơn

Cho các struct định nghĩa một DSLK đơn và một node của DSLK đó:

```
struct Node {
    int key;
    Node* pNext;
};

struct List {
    Node* pHead;
    Node* pTail;
};
```

Dùng (các) prototype cho trước, viết các hàm thực hiện các yêu cầu sau:

1. Tạo một Node từ một số nguyên `data` cho trước
   ```
   Node* createNode(int data);
   ```
2. Khởi tạo List từ một Node cho trước
   ```
   List* createList(Node* pNode);
   ```
3. Chèn một số nguyên `data` vào đầu List cho trước
   ```
   bool addHead(List*& L, int data);
   ```
4. Chèn một số nguyên `data` vào cuối List cho trước
   ```
   bool addTail(List*& L, int data);
   ```
5. Chèn một số nguyên `val` vào một vị trí `pos`
   ```
   bool addPos(List*& L, int val, int
       pos);
   ```
6. Chèn một số nguyên `data` vào trước vị trí của Node mang giá trị `val`
   ```
   bool addBefore(List*& L, int data,
       int val);
   ```
7. Chèn một số nguyên `data` vào sau vị trí của Node mang giá trị `val`
   ```
   bool addAfter(List*& L, int data,
       int val);
   ```
8. Xóa node đầu của List
   ```
   void removeHead(List*& L);
   ```
9. Xóa node cuối của List
   ```
   void removeTail(List*& L);
   ```
10. Xóa node ở vị trí `pos` trong List
    ```
    void removePos(List*& L, int pos);
    ```
11. Xóa tất cả node trong List
    ```
    void removeAll(List*& L);
    ```
12. Xóa node đứng trước node mang giá trị `val` trong List
    ```
    void removeBefore(List* L, int val);
    ```
13. Xóa node đứng sau node mang giá trị `val` trong List
    ```
    void removeAfter(List* L, int val);
    ```
14. In giá trị các node có trong list ra màn hình
    ```
    void printList(List* L);
    ```
15. Đếm số lượng node có trong list
    ```
    int countElements(List* L);
    ```
16. Đảo thứ tự các node trong list và lưu vào một list mới

```
List* reverseList(List* L);
```

17. Xóa các node sao cho list không có node nào trùng nhau

```
void removeDuplicate(List*& L);
```

18. Xóa các node có giá trị `val` trong list

```
bool removeElement(List*& L, int val);
```

1.2 DSLK đôi

Cho các struct định nghĩa DSLK đôi và một Node của DSLK đó:

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

struct DList {
    DNode* pHead;
    DNode* pTail;
};
```

Thực hiện các yêu cầu ở 1.1 với DSLK đôi trên.

2 Stack - Queue

Cho struct định nghĩa một Node của DSLK đơn:

```
struct Node {
    int key;
    Node* pNext;
};
```

Từ định nghĩa trên, cài đặt Stack và Queue bằng DSLK đơn, sau đó viết các hàm thực hiện các yêu cầu sau:

- Stack
    1. Khởi tạo Stack từ một `key` cho trước
    2. **Push** một `key` vào Stack
    3. **Pop** một Node ra khỏi Stack, trả về giá trị của Node này
    4. Đếm số lượng các Node trong Stack
    5. Kiểm tra Stack có rỗng hay không
- Queue
    1. Khởi tạo Queue từ một `key` cho trước
    2. **Enqueue** một `key` vào Queue
    3. **Dequeue** một Node ra khỏi Queue, trả về giá trị của Node này
    4. Đếm số lượng các Node trong Queue
    5. Kiểm tra Queue có rỗng hay không