REPORT BIG ASSIGNMENT
Chương 2

Họ tên: Nguyễn Anh Tuấn
MSSV: 20200400

Requirement:

Cài đặt thuật toán QuickSort khử đệ quy bằng danh sách liên kết đơn

Bài làm:

Code:

Khởi tạo cấu trúc cho Node

```
11  #include <iostream>
12
13  using namespace std;
14
    You, 12 minutes ago | 1 author (You)
15  struct Node {
16      int data;
17      Node* next;
18  };
```

Hàm thêm một node mới với data vào sau danh sách đã có

```
64  void addTail(Node** head, int data){
65      Node* node = new Node();
66      node->data = data;
67      node->next = NULL;
68      if (*head == NULL) {
69          *head = node;
70      } else {
71          Node* tmp = *head;
72          while (tmp->next != NULL) {
73              tmp = tmp->next;
74          }
75          tmp->next = node;
76      }
77  }
```

Hàm đếm số lượng phần tử/node trong toàn bộ danh sách

```
79  int count(Node** head){
80      int size = 0;
81      Node* tmp = *head;
82      while (tmp != NULL){
83          size++;
84          tmp = tmp->next;
85      }
86      return size;
87  }
```

Hàm trả về giá trị data tại vị trí pos của danh sách như a[i] trong array

```
89  int get(Node** head, int pos){
90      int cnt = 0;
91      int size = count(head);
92      if ( pos >= size ) throw "Out of bound";
93      for ( Node* tmp = *head; cnt <= pos; tmp = tmp->next){
94          if ( cnt == pos) return tmp->data;
95          cnt++;
96      }
97      return -1;
98  }
```

Hàm đổi data của 2 node tại vị trí I và j trong danh sách

```
100  void swapNodeAtIndex(Node** head, int i, int j) {
101      if (*head == nullptr || i == j) {
102          return;
103      }
104      Node* a = nullptr;
105      Node* b = nullptr;
106      Node* tmp = *head;
107      int index = 0;
108      while (tmp != nullptr) {
109          if (index == i) {
110              a = tmp;
111          } else if (index == j) {
112              b = tmp;
113          }
114          tmp = tmp->next;
115          ++index;
116      }
117      if (a == nullptr || b == nullptr) {
118          return;
119      }
120      int temp = a->data;
121      a->data = b->data;
122      b->data = temp;
123  }
```

Hàm in node ra màn hình

```
153  void logNode(Node* head){
154      Node* tmp = head;
155      while (tmp != NULL){
156          cout << tmp->data << " ";
157          tmp = tmp->next;
158      }
159      cout << endl;
160  }
```

Khởi tạo cấu trúc cho Stack và các hàm:
+ push: thêm node với data vào stack
+ pop: xoá 1 phần tử node ra khỏi stack theo luồng LIFO
+ topNode: trả về node đầu tiên theo luồng LIFO
+ isEmpty: hàm kiểm tra xem stack có rỗng hay không
+ print: in stack ra màn hình

```
20  class Stack {
21  public:
22      Node* top;
23
24  public:
25      Stack() {
26          top = nullptr;
27      }
28      void push(int value) {
29          Node* newNode = new Node;
30          newNode->data = value;
31          newNode->next = top;
32          top = newNode;
33      }
34      void pop() {
35          if (isEmpty()) {
36              throw "Stack is empty!";
37          }
38          Node* temp = top;
39          top = top->next;
40          delete temp;
41      }
42      Node* topNode() {
43          if (isEmpty()) {
44              throw "Stack is empty!";
45          }
46          Node* tmp = new Node();
47          tmp->data = top->data;
48          tmp->next = NULL;
49          return tmp;
50      }
51      bool isEmpty() {
52          return top == nullptr;
53      }
54      void print() {
55          Node* tmp = top;
56          while (tmp != nullptr) {
57              cout << tmp->data << " ";
58              tmp = tmp->next;
59          }
60          cout << endl;
61      }
62  };
```

Hàm quicksort khử đệ quy sử dụng cho danh sách liên kết cùng với sự hỗ trợ của các hàm trên

```
125  void QuickSort(Node** head, int l, int r){
126      Node* a = *head;
127      Stack sl, sr;
128      sl.push(l); sr.push(r);
129      while (!sl.isEmpty()){
130          l = sl.topNode()->data; sl.pop();
131          r = sr.topNode()->data; sr.pop();
132          int mid = (l+r)/2;
133          int x = get(&a,mid);
134          int i = l, j = r;
135          while (i <= j){
136              while(get(&a,i) < x) i++;
137              while(get(&a,j) > x) j--;
138              if (i <= j){
139                  if ( i < j ) swapNodeAtIndex(&a,i,j);
140                  i++; j--;
141              }
142          }
143          if (l < j) {
144              sl.push(l); sr.push(j);
145          }
146          if (i < r) {
147              sl.push(i); sr.push(r);
148          }
149      }
150      return;
151  }
```

Kết quả chụp màn hình :

```
162  int main() {
163      Node* head = NULL;
164      addTail(&head, 5);
165      addTail(&head, 15);
166      addTail(&head, 18);
167      addTail(&head, 43);
168      addTail(&head, 28);
169      addTail(&head, 90);
170      addTail(&head, 35);
171      addTail(&head, 1);
172      int n = count(&head);
173
174      cout << "Danh sach lien ket truoc khi sap xep: \n";
175      logNode(head);
176      QuickSort(&head,0,n-1);
177      cout << "Danh sach lien ket sau khi sap xep: \n";
178      logNode(head);
179
180      return 0;
181  }
```

```
(base) macad@nganhhtuann-3 output % ./"quickSortList"
Danh sach lien ket truoc khi sap xep:
5 15 18 43 28 90 35 1
Danh sach lien ket sau khi sap xep:
1 5 15 18 28 35 43 90
```

Nhận xét:

Cài đặt thuật toán phức tạp

Giảm thiểu độ phức tạp so với quicksort dùng đệ quy

Hàm quicksort dùng stack lưu trữ index bắt đầu và cuối của các danh sách con trong danh sách liên kết thay vì sử dụng đệ quy -> không cần lưu trữ biến cục bộ và địa chỉ trả về trên stack đỡ tràn bộ nhớ và dễ dàng tối ưu hoá hiệu quả hơn.