Midterm Test (75 minutes)

1 Danh sách liên kết

Cho một danh sách liên kết đơn được định nghĩa như sau:

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

```
struct List{
    NODE* p_head;

};
```

1. (2pt) Viết hàm chèn một số nguyên vào vị trí bất kì của `List` cho trước, nếu vị trí lớn hơn chiều dài `List`, chèn vào cuối:

   - `List* addPos(List* L, int data, int pos)`

   VD:

   Input: L = 1 -> 2 -> 3, data = 4, pos = 2

   Output: 1 -> 2 -> 4 -> 3

2. (1pt) Viết hàm xóa vị trí pos tính từ cuối chuỗi

   - `List* removeFromEnd(List* L, int pos)`

   VD:

   Input: L = 1 -> 2 -> 4 -> 3, pos = 1

   Output: 1 -> 2 -> 3 (Tính từ cuối `List`, vị trí 1 là 4)

3. (1pt) Viết hàm tráo đổi 2 node ở vị trí pos1(tính từ đầu `List`) và post2 (tính từ cuối `List`) (hoán đổi node, không chỉ hoán đổi giá trị)

   - `List* swapNode(List* L, int pos1, pos2)`

   VD:

   Input: L = 1 -> 2 -> 4 -> 3 -> 5, pos1 = 1, pos2 = 2

   Output: 1 -> 4 -> 2 -> 3 -> 5 (Tính từ đầu `List`, vị trí pos1 là node có giá trị 2, tính từ cuối `List`, vị trí pos2 là node có giá trị 4)

2 Cây AVL

Mỗi Node của một cây AVL được định nghĩa như sau:

```
struct AVLNode{
    int key;
    NODE* p_left;
    NODE* p_right;
};
```

Sinh viên cần thực hiện cài đặt các hàm sau:

1. (1pt) Khởi tạo một `NODE` từ một giá trị cho trước:
   - `NODE* createAVLNode(int data)`
2. (0.5pt) Duyệt tiền thứ tự:
   - `void NLR(AVLNODE* pRoot)`
3. (0.5pt) Tìm và trả về một `NODE` với giá trị cho trước từ một cây nhị phân tìm kiếm:
   - `NODE* Search(AVLNODE* pRoot, int x)`
4. (1.5pt) Thêm một `AVLNODE` với giá trị cho trước vào cây AVL
   - `void Insert(AVLNODE* &pRoot, int x)`
5. (1.5pt) Xoá một `AVLNODE` với giá trị cho trước từ một cây nhị phân tìm kiếm
   - `void Remove(AVLNODE* &pRoot, int x)`
6. (1pt) Với số k cho trước, Kiểm tra có tồn tại cặp a, b trong cây AVL sao cho a + b = k hay không sao cho độ phức tạp của giải thuật là O(n). (Biết cây AVL ít hơn 100 Node)
   - `bool checkExist(AVLNODE* pRoot, int k)`