KIỂM TRA CUỐI KỲ - 20CNTN

Thời gian: 60'

Quy định

- Nộp file *"MSSV.rar/zip"*.
- Trong thư mục *"MSSV"* chứa các file *"1.cpp"*, *"2a.cpp"*, *"2b.cpp"* lần lượt là bài làm của các câu **1.** , **2a.** và **2b.**.
- Sinh viên được quyền viết thêm các hàm bổ trợ nhưng không sửa prototype của các hàm cho sẵn.

Nội dung

Câu 1 (5 điểm)

Cho định nghĩa cấu trúc thể hiện thông tin của một node trong cây AVL như sau:

```
struct AVLNode {
    int p, q, r;
    AVLNode* pLeft;
    AVLNode* pRight;
};
```

Cấu trúc này thể hiện các hỗn số có dạng $p\frac{q}{r}$. Hãy cài đặt các hàm sau:

- Hàm `Insert` cho phép thêm một node (cần được rút gọn) vào cây AVL.
  `void Insert(AVLNode* &root, int pInp, int qInp, int rInp)`
- Hàm `Remove` cho phép xóa một node có giá trị `(p, q, r)` bằng với cặp tham số `(qInp, pInp, rInp)` được truyền vào hàm.
  `void Remove(AVLNode* &root, int pInp, int qInp, int rInp)`
- Hàm `CreateTree` cho phép tạo cây AVL từ **mảng** các giá trị `(p, q)` được truyền vào hàm.
  `AVLNode* CreateTree(vector <int> pList, vector <int> qList, vector <int> rList)`
- Hàm `LevelOrder` cho phép duyệt cây AVL theo mức, giá trị mỗi node in ra có định dạng `(p, q, r)`.
  `void LevelOrder(AVLNode* root)`

Câu 2

a. *(2 điểm)* Một đồ thị có 5 đỉnh bất kỳ được xem là đồ thị hình chiếc nơ nếu nó thỏa các đặc điểm sau:

- Không có khuyên.
- Có hình dạng:

Hãy cài đặt hàm `bool isBowtie(vector<vector<bool>> adj)` nhận input là một ma trận kề, cho biết đồ thị tương ứng có phải là đồ thị hình chiếc nơ hay không.

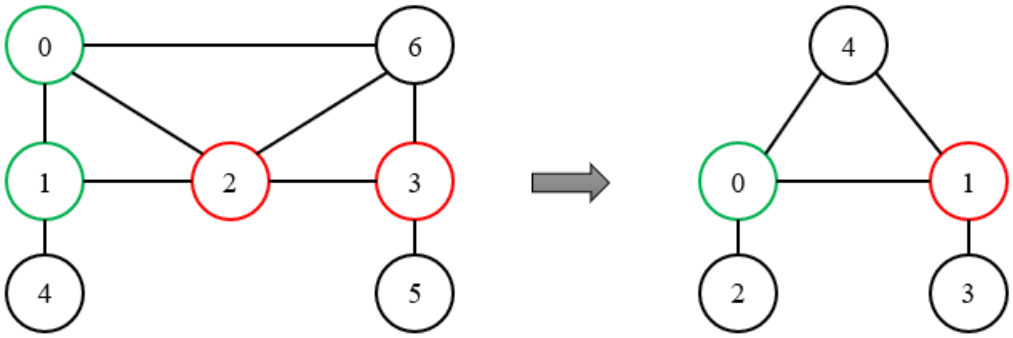
Ví dụ:

Input					Output
1	0	1	0	0	True
0	1	1	0	0	
1	1	0	1	1	
0	0	1	0	1	
0	0	1	1	0	

b. (3 điểm) Xét một đồ thị vô hướng G cho trước. Hãy viết hàm nhận input là một ma trận kề, gộp các cặp đỉnh liên tiếp của đồ thị G để tạo ra đồ thị mới `G'`, quy định như sau:

- Nếu 2 đỉnh v và $v+1$ có cạnh nối, sẽ được gộp thành 1 đỉnh mới đánh số v.
- Sau khi gộp cặp đỉnh v và $v+1$, các đỉnh i với $i > v$ được đổi thành $i-1$.

  Ví dụ:



Input						
0	1	1	0	0	0	1
1	0	1	0	1	0	0
1	1	0	1	0	0	1
0	0	1	0	0	1	1
0	1	0	0	0	0	0
0	0	0	1	0	0	0
1	0	1	1	0	0	0

Output				
0	1	1	0	1
1	0	0	1	1
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0

Prototype: `vector <vector <int>> solution (vector <vector <int>> adj)`

HẾT