[Cây NPTK] Các khóa có chi phí tìm kiếm cao nhất.

Hãy hoàn thành hàm **MaxFindingCost(TREE)** để in ra các khóa có chi phí tìm kiếm lớn nhất trên cây nhị phân tìm kiếm theo thứ tự tăng dần. Các khóa được in ra trên 1 dòng và cách nhau bằng một khoảng trắng.

Lưu ý:

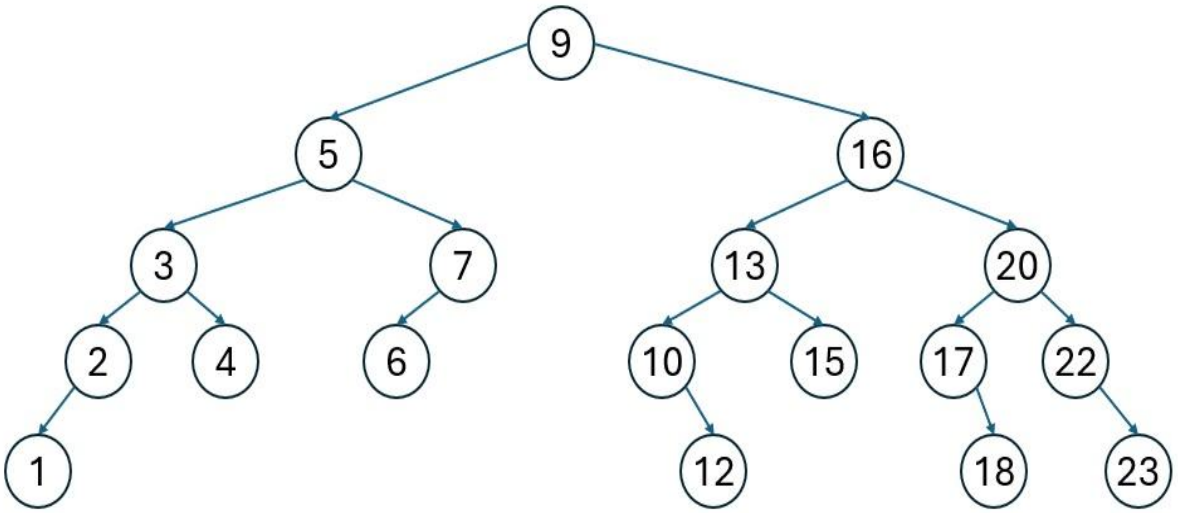
1) Chỉ cài đặt hàm **MaxFindingCost** và các hàm liên quan (nếu cần) để in ra các khóa theo yêu cầu của đề bài.

2) Kiểu dữ liệu và một số hàm liên quan đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng.

3) Đầu vào và đầu ra đã được xử lý sẵn và phù hợp với định dạng nhập/xuất.

Ví dụ: với cây nhị phân tìm kiếm như bên dưới, thì kết quả là:

1 12 18 23



[Cây NPTK] Kiểm tra cây nhị phân tìm kiếm.

Hãy hoàn thành hàm **bool isBST(TREE)** kiểm tra một cây nhị phân có phải là cây nhị phân tìm kiếm hay không. Nếu là cây nhị phân tìm kiếm, hàm sẽ trả về **true**, ngược lại là **false**. Biết rằng cây rỗng vẫn là một cây nhị phân tìm kiếm.

Lưu ý:

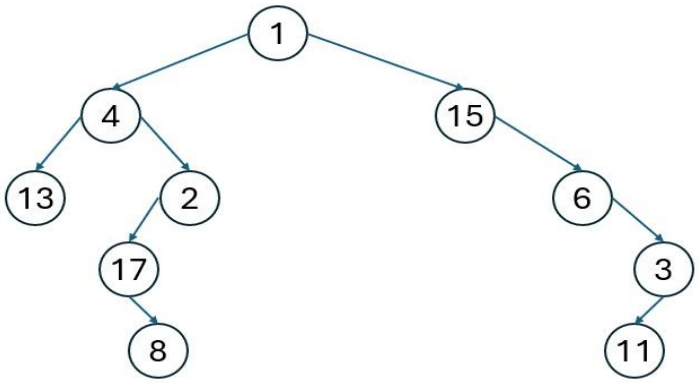
1) Chỉ cài đặt hàm **isBST(TREE)** và các hàm liên quan đến nó, để kiểm tra cây nhị phân tìm kiếm.

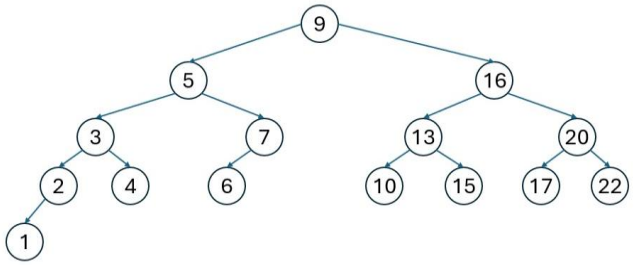
2) Kiểu dữ liệu và một số hàm liên quan đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng.

3) Đầu vào và đầu ra đã được xử lý sẵn và phù hợp với định dạng nhập/xuất.

Ví dụ: với hai cây bên dưới:

Input	Output
10 1 4 13 2 17 8 15 6 3 11 13 4 17 8 2 1 15 6 11 3	NO
15 9 5 3 2 1 4 7 6 16 13 10 15 20 17 22 1 2 3 4 5 6 7 9 10 13 15 16 17 20 22	YES





[Cây NPTK] Tạo cây lưu trữ thông tin học sinh.

Viết chương trình nhập vào n thông tin học sinh và tạo cây nhị phân tìm kiếm từ các thông tin nhập vào. Sau đó, in ra kết quả duyệt LNR của cây nhị phân tìm kiếm vừa tạo được. Biết: - Thứ tự thêm các nút vào cây là thứ tự nhập thông tin học sinh. - Thứ tự trên cây nhị phân tìm kiếm là thứ tự tăng dần của trường Ten học sinh. Nếu trường Ten trùng nhau thì xét thứ tự tăng dần của trường Hodem. - Nếu thông tin Ten và Hodem đã có trên cây thì không thêm vào nữa.

Đầu vào:

- Số nguyên dương n, cho biết có n thông tin học sinh.
- n khối kế tiếp, mỗi khối là toàn bộ thông tin của một học sinh, được đọc bằng hàm **InputElement(Hocsinh &)** đã được cài đặt sẵn.

Đầu ra: Kết quả duyệt LNR của cây nhị phân tìm kiếm vừa được tạo, gồm k dòng. Mỗi dòng là

một thông tin của học sinh. Hãy dùng hàm **OutputElement** để kết xuất thông tin của một học sinh(đã kèm theo dấu xuống dòng).

Lưu ý: bài này không được dùng mảng, vector và không sử dụng thư viện khác ngoài iostream.

Input	Output
6 Dinh Van Phuong 1 23 10 2006 10 7 6.5 8.5 10 9 Le Hoang Quan 1 25 8 2006 6.5 9 10 8.5 10 8 Le Tuan Anh 1 29 11 2006 10 9 9 8.5 7 6 Le Tuan Anh 1 10 12 2006 6 7 8.5 7.5 9 10 Nguyen The Nhu Quynh 0 20 7 2006 8 6.5 7 8.5 10 8 Le Tuan Anh 1 4 11 2005 8.5 8.5 10 6.5 7.5 7	Le Tuan Anh 1 29 11 2006 10 9 9 8.5 7 6 Dinh Van Phuong 1 23 10 2006 10 7 6.5 8.5 10 9 Le Hoang Quan 1 25 8 2006 6.5 9 10 8.5 10 8 Nguyen The Nhu Quynh 0 20 7 2006 8 6.5 7 8.5 10 8

[Cây NPTK] Số bước tìm khóa.

Hãy hoàn thành hàm **int ComparisonCount(TREE, int)** để đếm số phép so sánh để tìm thấy khóa số nguyên trên cây nhị phân tìm kiếm.

Lưu ý:

1) Chỉ cài đặt hàm **ComparisonCount**.

2) Kiểu dữ liệu và một số hàm liên quan đã được cài đặt sẵn. Sinh viên có thể đọc để sử dụng.

3) Đầu vào và đầu ra đã được xử lý sẵn và phù hợp với định dạng nhập/xuất.

Ví dụ: với cây nhị phân tìm kiếm như bên dưới, thì:

- Kết quả tìm khóa 1 là 5.

- Kết quả tìm khóa 13 là 3.
- Kết quả tìm khóa 22 là 4.