Μέρος 1: Συνδεδεμένες λίστες

Έστω μία υλοποίηση απλά συνδεδεμένης λίστας, οι κόμβοι της οποίας έχουν τον τύπο:

```
struct list_node
{
  int data;
  list_node *next;
};
```

Ζητείται να υλοποιήσετε τις συναρτήσεις με επικεφαλίδες:

```
int sum(list_node *n);
bool is_arithmetic_progression(list_node *n);
```

όπου:

- **sum(l)**: Αθροίζει τις τιμές των κόμβων της λίστας **l**.
- **is_arithmetic_progression(l)**: Επιστρέφει **true** αν οι τιμές που περιέχονται στη λίστα **l** ορίζουν αριθμητική πρόοδο, διαφορετικά **false**. Για μία κενή λίστα ή για μία λίστα με ένα μόνο στοιχείο, η συνάρτησή σας πρέπει να επιστρέφει **true**.

Μέρος 2: Δυαδικά δένδρα

Έστω μία υλοποίηση δυαδικού δένδρου, οι κόμβοι του οποίου έχουν τον τύπο:

```
struct tree_node
{
  int data;
  tree_node *left, *right;
};
```

Ζητείται να υλοποιήσετε τις συναρτήσεις με επικεφαλίδες:

```
int count_smaller_than(tree_node *t, int x);
bool find(tree_node *t, int x);
bool is_bst(tree_node *t);
```

όπου:

- **count_smaller_than(t, x)**: Επιστρέφει το πλήθος των στοιχείων του δένδρου **t** που είναι (αυστηρά) *μεγαλύτερα* του **x**.
- **find(t, x)**: Επιστρέφει **true** αν το στοιχείο **x** υπάρχει στο δένδρο **t**, διαφορετικά **false**. Το δένδρο αυτό ΔΕΝ είναι υποχρεωτικά δένδρο δυαδικής αναζήτησης (BST) και μπορεί να είναι κενό.

- **is_bst(t)**: Επιστρέφει **true** αν το δένδρο **t** είναι δένδρο δυαδικής αναζήτησης, διαφορετικά **false**. Υποθέστε ότι οι τιμές των κόμβων του δένδρου έχουν απόλυτη τιμή που δεν υπερβαίνει το 1.000.000 και ότι δεν υπάρχουν διπλότυπα.

Περιορισμοί

- **Όριο χρόνου εκτέλεσης:** 1 sec.
- **Όριο μνήμης:** 128 MB.