Question **1**

Correct

Mark 100.00 out of 100.00

| | |
|---|---|
| Time limit | 1 s |
| Memory limit | 64 MB |

**Nama File:** bst.c

Di laboratorium Gro yang ramai, para Nimons sedang berlomba untuk mendapatkan gadget terbaru: "Banana Sorting 5000". Mesin ini dapat mengurutkan pisang berdasarkan panjangnya secara otomatis. Namun, mesin ini rusak dan Gro membutuhkan bantuan para Nimons untuk tetap dapat mengelola persediaan pisang mereka.

Dave, seorang Nimon yang cukup terampil dengan perkakas (walaupun seringkali berakhir dengan kekacauan), mengusulkan untuk membuat sistem penyimpanan pisang digital. Ia membayangkan setiap pisang akan direpresentasikan oleh panjangnya (dalam satuan Nimon Fingers), dan mereka akan disimpan dalam sebuah struktur data khusus yang memungkinkan pencarian, penambahan, dan identifikasi pisang terpendek dengan cepat. Dave mendengar tentang konsep **Binary Search Tree (BST)** dan merasa ini sangat cocok untuk tugas mereka.

Pop, Kebin, dan Stewart setuju untuk membantu. Mereka perlu mengimplementasikan ADT untuk sistem penyimpanan pisang ini dalam bahasa C. Mereka membutuhkan fungsi untuk menambahkan pisang baru (berdasarkan panjangnya), mencari apakah pisang dengan panjang tertentu sudah ada, dan dengan cepat menemukan pisang dengan panjang terkecil di antara semua pisang yang tersimpan.

Tantangan bagi para Nimons adalah mengimplementasikan header ADT Binary Search Tree ini ke dalam file `bst.c`. Mereka sedikit kesulitan membayangkan bagaimana cara menyusun 'node' pisang ini sehingga pencarian dan penemuan nilai minimum dapat dilakukan secara efisien. Mereka juga perlu memastikan tidak ada duplikasi panjang pisang yang tersimpan.

Bantulah Dave, Pop, Kebin, dan Stewart untuk menyelesaikan ADT ini dengan melengkapi file bst.h. Kumpulkan hanya file `bst.c`. Gunakan file main.c untuk melakukan pengujian terhadap implementasi yang kalian buat. Output yang diharapkan adalah sebagai berikut.

Melakukan pencarian node:

Node dengan nilai 40 ditemukan (count: 1).

Node dengan nilai 90 tidak ditemukan.

Mencari nilai minimum:

Nilai minimum dalam BST adalah: 20 (count: 1)

Lampiran:
boolean.h

C

bst.c

## Score: 100

## Blackbox

### Score: 100

### Verdict: Accepted

### Evaluator: Exact

| No | Score | Verdict | Description |
|---|---|---|---|
| 1 | 20 | Accepted | 0.00 sec, 1.56 MB |
| 2 | 20 | Accepted | 0.00 sec, 1.61 MB |
| 3 | 20 | Accepted | 0.00 sec, 1.54 MB |
| 4 | 20 | Accepted | 0.00 sec, 1.50 MB |
| 5 | 20 | Accepted | 0.00 sec, 1.46 MB |

Question **2**

Correct

Mark 100.00 out of 100.00

| | |
|---|---|
| Time limit | 1 s |
| Memory limit | 64 MB |

**Nama File**: bintree_traversal.c

Gro telah diculik oleh sosok penjahat misterius yang sangat cerdik. Untungnya, sang penjahat meninggalkan jejak dalam bentuk pohon biner seimbang (**Balanced Binary Tree**) yang berisi kode lokasi rahasia menuju markas tempat Gro disembunyikan.

Tiga Nimons terhebat, yaitu Kebin, Pop, dan Stewart ditugaskan untuk menyelamatkan Gro. Namun, ketiganya memiliki gaya membaca jejak yang berbeda:

- Kebin selalu membaca pesan dengan **PreOrder traversal**
- Pop percaya urutan yang benar adalah dengan **InOrder traversal**
- Stewart lebih suka mengandalkan **PostOrder traversal**

Karena tidak ingin salah langkah dan kehilangan jejak, mereka sepakat untuk mencetak ketiga rute traversal agar dapat membandingkan dan memilih jalur yang paling mungkin menuju markas penjahat.

Sebagai orang yang baik hati, kamu diminta untuk membantu ketiga Nimons mengimplementasikan **bintree_traversal.h** yang dapat menampilkan isi pohon dalam urutan PreOrder, InOrder, dan PostOrder, sekaligus mengonversi tiap urutan traversal tersebut menjadi sebuah NodeList lalu dicetak.

Anda hanya diminta mengumpulkan `bintree_traversal.c` nya saja. Dalam pengerjaan, Anda disarankan untuk memanfaatkan file header dari soal sebelumnya, yaitu **bst.h**. Selain itu, tersedia pula file **boolean.h** yang dapat digunakan bila diperlukan dalam implementasi Anda.

**Gunakan main.c untuk menguji program yang Anda buat.**

**Keluaran yang diharapkan dari main.c tersebut:**

```
Traversal PreOrder: 8 3 2 1 6 4 7 10 14 13
Traversal InOrder: 1 2 3 4 6 7 8 10 13 14
Traversal PostOrder: 1 2 4 7 6 3 13 14 10 8

List PreOrder: [8] -> [3] -> [2] -> [1] -> [6] -> [4] -> [7] -> [10] -> [14] -> [13] -> FINISH
List InOrder: [1] -> [2] -> [3] -> [4] -> [6] -> [7] -> [8] -> [10] -> [13] -> [14] -> FINISH
List PostOrder: [1] -> [2] -> [4] -> [7] -> [6] -> [3] -> [13] -> [14] -> [10] -> [8] -> FINISH
```

C

bintree_traversal.c

## Score: 100

## Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

| No | Score | Verdict | Description |
|---|---|---|---|
| 1 | 10 | Accepted | 0.00 sec, 1.67 MB |
| 2 | 10 | Accepted | 0.00 sec, 1.64 MB |
| 3 | 10 | Accepted | 0.00 sec, 1.67 MB |
| 4 | 10 | Accepted | 0.00 sec, 1.67 MB |
| 5 | 10 | Accepted | 0.00 sec, 1.61 MB |
| 6 | 10 | Accepted | 0.00 sec, 1.71 MB |
| 7 | 10 | Accepted | 0.00 sec, 1.65 MB |
| 8 | 10 | Accepted | 0.00 sec, 1.55 MB |
| 9 | 10 | Accepted | 0.00 sec, 1.65 MB |
| 10 | 10 | Accepted | 0.00 sec, 1.72 MB |

◄ Post Praktikum 8 - K3 & K4

Jump to...

Praktikum 9 - K3 & K4 ►