(/)

You have a captain's log due before 2020-08-02 (in 1 day)! Log it now!
(/captain_logs/40904/edit)

You are already signed in.

0x02. Heap Insert

Specializations - Interview Preparation — Algorithms

by Alexa Orrico, Software Engineer at Holberton School

Ongoing project - started 07-29-2020, must end by 08-05-2020 (in 3 days) - you're done with 50% of tasks.

Checker was released at 08-01-2020 12:00 PM

QA review fully automated.

binary trees c heaps queues

Requirements

General

- Allowed editors: `vi, vim, emacs`
- All your files will be compiled on Ubuntu 14.04 LTS
- Your programs and functions will be compiled with `gcc 4.8.4` using the flags `-Wall -Werror -Wextra` and `-pedantic`
- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the `Betty` style. It will be checked using betty-style.pl (https://github.com/holbertonschool/Betty/blob/master/betty-style.pl) and betty-doc.pl (https://github.com/holbertonschool/Betty/blob/master/betty-doc.pl)
- You are not allowed to use global variables
- No more than 5 functions per file
- You are allowed to use the standard library
- In the following examples, the `main.c` files are shown as examples. You can use them to test your functions, but you don't have to push them to your repo (if you do we won't take them into account).

We will use our own `main.c` files at compilation. Our `main.c` files might be different from the one shown in the examples
- The prototypes of all your functions should be included in your header file called `binary_trees.h`
- Don't forget to push your header file
- All your header files should be include guarded

More Info

Data structures

Please use the following data structures and types for binary trees. Don't forget to include them in your header file.

Basic Binary Tree

```
/**
 * struct binary_tree_s - Binary tree node
 *
 * @n: Integer stored in the node
 * @parent: Pointer to the parent node
 * @left: Pointer to the left child node
 * @right: Pointer to the right child node
 */
struct binary_tree_s
{
    int n;
    struct binary_tree_s *parent;
    struct binary_tree_s *left;
    struct binary_tree_s *right;
};
```

Max Binary Heap

```
typedef struct binary_tree_s heap_t;
```

Print function

To match the examples in the tasks, you are given this function (https://github.com/holbertonschool/0x1C.c)

This function is used only for visualisation purpose. You don't have to push it to your repo. It may not be used during the correction

Tasks

0. New node mandatory

Done!

Help

Write a function that creates a binary tree node:

- Prototype: `binary_tree_t *binary_tree_node(binary_tree_t *parent, int value);`
- `parent` is a pointer to the parent node of the node to create
- `value` is the value to put in the new node
- When created, a node does not have any children
- Your function must return a pointer to the new node, or `NULL` on failure

```
alex@/tmp/binary_trees$ cat 0-main.c
#include <stdlib.h>
#include "binary_trees.h"

/**
 * main - Entry point
 *
 * Return: Always 0 (Success)
 */
int main(void)
{
    binary_tree_t *root;

    root = binary_tree_node(NULL, 98);

    root->left = binary_tree_node(root, 12);
    root->left->left = binary_tree_node(root->left, 6);
    root->left->right = binary_tree_node(root->left, 16);

    root->right = binary_tree_node(root, 402);
    root->right->left = binary_tree_node(root->right, 256);
    root->right->right = binary_tree_node(root->right, 512);

    binary_tree_print(root);
    return (0);
}
alex@/tmp/binary_trees$ gcc -Wall -Wextra -Werror -pedantic binary_tree_print.c 0-main.
c 0-binary_tree_node.c -o 0-node
alex@/tmp/binary_trees$ ./0-node
       .-------(098)-------.
  .--(012)--.         .--(402)--.
(006)     (016)     (256)     (512)
alex@/tmp/binary_trees$
```

Repo:

- GitHub repository: `holbertonschool-interview`
- Directory: `0x02-heap_insert`
- File: `0-binary_tree_node.c`

Check your code?

1. Heap - Insert mandatory

Done?

Help

Write a function that inserts a value into a Max Binary Heap:

- Prototype: `heap_t *heap_insert(heap_t **root, int value)`
- `root` is a double pointer to the root node of the Heap
- `value` is the value store in the node to be inserted
- Your function must return a pointer to the inserted node, or `NULL` on failure
- If the address stored in `root` is `NULL`, the created node must become the root node.
- You have to respect a `Max Heap` ordering
- You are allowed to have up to `6` functions in your file

Your file `0-binary_tree_node.c` will be compiled during the correction

```
alex@/tmp/binary_trees$ cat 1-main.c
#include <stdlib.h>
#include <stdio.h>
#include "binary_trees.h"

/**
 * main - Entry point
 *
 * Return: 0 on success, error code on failure
 */
int main(void)
{
    heap_t *root;
    heap_t *node;

    root = NULL;
    node = heap_insert(&root, 98);
    printf("Inserted: %d\n", node->n);
    binary_tree_print(root);
    node = heap_insert(&root, 402);
    printf("\nInserted: %d\n", node->n);
    binary_tree_print(root);
    node = heap_insert(&root, 12);
    printf("\nInserted: %d\n", node->n);
    binary_tree_print(root);
    node = heap_insert(&root, 46);
    printf("\nInserted: %d\n", node->n);
    binary_tree_print(root);
    node = heap_insert(&root, 128);
    printf("\nInserted: %d\n", node->n);
    binary_tree_print(root);
    node = heap_insert(&root, 256);
    printf("\nInserted: %d\n", node->n);
    binary_tree_print(root);
    node = heap_insert(&root, 512);
    printf("\nInserted: %d\n", node->n);
    binary_tree_print(root);
    node = heap_insert(&root, 50);
    printf("\nInserted: %d\n", node->n);
    binary_tree_print(root);
    return (0);
}

alex@/tmp/binary_trees$ gcc -Wall -Wextra -Werror -pedantic binary_tree_print.c 1-main.
c 1-heap_insert.c 0-binary_tree_node.c -o 1-heap_insert
alex@/tmp/binary_trees$ ./1-heap_insert
Inserted: 98
(098)

Inserted: 402
  .--(402)
(098)
```

```
Inserted: 12
  .--(402)--.
(098)     (012)

Inserted: 46
       .--(402)--.
  .--(098)     (012)
(046)

Inserted: 128
       .-------(402)--.
  .--(128)--.       (012)
(046)     (098)

Inserted: 256
       .-------(402)-------.
  .--(128)--.         .--(256)
(046)     (098)     (012)

Inserted: 512
       .-------(512)-------.
  .--(128)--.         .--(402)--.
(046)     (098)     (012)     (256)

Inserted: 50
            .-------(512)-------.
       .--(128)--.         .--(402)--.
  .--(050)     (098)     (012)     (256)
(046)
alex@/tmp/binary_trees$
```

Repo:

- GitHub repository: `holbertonschool-interview`
- Directory: `0x02-heap_insert`
- File: `1-heap_insert.c, 0-binary_tree_node.c`

Check your code?