Skilaverkefni VI
Tvíundaleitartré / Binary Search Tree

Lýsing á íslensku

Í þessu verkefni eigið þið að útfæra aðgerðir í `BinarySearchTree` klasanum sem við höfum fjallað um í fyrirlestri, þ.e. `maxNode()`, `minNode()`, `insert()` og `remove()`. Ofangreindar aðgerðir eru `public` í `BinarySearchTree` en þið þurfið jafnframt að útfæra `private` hjálparföll sem þessar aðgerðir nota (sjá `binarysearchtree.h`)

Smiðir og föllin `find()` og `findAt()` eru gefin (sjá `binarysearchtree.cpp`).

Helsta flækjan í þessu verkefni er útfærslan á `remove()` aðgerðinni, sbr. umfjöllun okkar í fyrirlestri. Þið eigið að láta `remove()` fallið kalla á `removeAt()`. Seinna fallið leitar síðan að hnútnum sem á að fjarlægja og kallar á `removeNode()` fyrir þann hnút. `removeNode()` er þá fallið sem sér um tilfellin þrjú (sem við ræddum í fyrirlestri) og eitt þeirra ("case 3") kallar á `processLeftmost()`.

Athugið að `BinarySearchTree` klasinn erfir frá `BinaryTree` klasanum sem þið útfærðuð í síðustu viku (æfingaverkefni í viku 8). Notið því útfærsluna á `BinaryTree (binarytree.cpp)` í þessu verkefni. Ef þið útfærðuð ekki `BinaryTree` klasann þá getið þið notað lausnina á honum sem verður aðgengileg í Myschool eða (sem þið lærið meira af) einfaldlega útfært `binarytree.cpp` samhliða þessu verkefni.

Eftirtaldar skrár (að hluta til eða að fullu) fáið þið gefnar: `binarynode.h`, `binarytree.h`, `binarytree.cpp`, `binarysearchtree.h`, `binarysearchtree.cpp` og `main.cpp`. Skráin sem þið þurfið að útfæra er þá `binarysearchtree.ccp` (og hugsanlega `binarytree.cpp`)

Skilið öllum ofangreindum skrám í Mooshak í einni .zip skrá.

English description

In this project, you need to implement operations in the `BinarySearchTree` class discussed in class, i.e. `maxNode()`, `minNode()`, `insert()` and `remove()`. The above methods are `public` in `BinarySearchTree`, but you also need to implement `private` helper functions used by these methods (see `binarysearchtree.h`)

Constructors and the methods `find()` and `findAt()` are given (see `binarysearchtree.cpp`).

The main complication in this project is the implementation of the `remove()` method, which we discussed in class. You should make `remove()` call `removeAt()`. The latter method searches for the node to be removed and calls `removeNode()` for that node. Thus, `removeNode()` is the function

which takes care of the three cases (discussed in class) and one of them ("case 3") calls `processLeftmost()`.

Note that the `BinarySearchTree` class inherits from the `BinaryTree` class, which you implemented in the last project (week 8). Therefore, you should use your implementation of `BinaryTree (binarytree.cpp)` in this project. If you did not implement the `BinaryTree` class, then you can use a solution which will be provided in Myschool, or simply (which facilitates your learning) implement `binarytree.cpp` as part of this current project.

The following files (partly or fully) are given: `binarynode.h`, `binarytree.h`, `binarytree.cpp`, `binarysearchtree.h`, `binarysearchtree.cpp` and `main.cpp`. You need to implement `binarysearchtree.ccp` (and possibly `binarytree.cpp`).

You need to hand in all of the above files in Mooshak as a single .zip file

Rétt úttak úr gefnu aðalforriti / Correct output from the given main function:

```
Create a Binary Search Tree (BST):
The height of the BST is: 4
The number of nodes in the BST is: 11
The item at the root is now: 15
Inorder: 2 3 4 6 7 9 13 15 17 18 20
Found 7 in the BST
Largest item in the BST is: 20
Smallest item in the BST is: 2
Inorder after first remove: 2 3 6 7 9 13 15 17 18 20
Inorder after second remove: 2 3 6 7 9 13 17 18 20
The item at the root is now: 17
Inorder after third remove: 2 3 6 9 13 17 18 20
Inorder after fourth remove: 2 3 6 9 13 17 18 20
```