Ef þú átt í vandræðum með að finna tiltekna stafi á lyklaborðinu þínu í prófinu þá eru hér nokkrir til að afrita / *If you have problems finding specific characters on your keyboard, here are a few to copy*: ( ) { } [ ] < > = & | / \ ! "

1. BinaryTree (50%)

Lýsing á íslensku

Í þessu verkefni eigið þið að útfæra klasann `BinaryTree`. Skilin (`binarytree.h`) eru gefin in þið þurfið að útfæra það sem á vantar í `binarytree.cpp`.

Hnútar tvíundatrésins, sem innihalda `char` gögn, eru geymdir í `vector` (en ekki í tengdum lista). Með því að nota `vector` þá er sérhver hnútur `v` í tvíundatrénu `T` með tiltekið númer sem fæst með fallinu `f(v)`:

- Ef `v` er rótin á `T`, þá er `f(v) = 1`
- Ef `v` er vinstra barn hnútar `u`, þá er `f(v) = 2*f(u)`
- Ef `v` er hægra barn hnútar `u`, þá er `f(v) = 2*f(u) + 1`

`f(v)` skilar þá vísi inn í undirliggjandi `vector` þar sem hnúturinn `v` er geymdur. Rótin er t.d. geymd í hólfi nr. 1 í undirliggjandi `vector`. Takið eftir því að fyrsta hólfið (nr. 0) í undirliggjandi `vector` er ekki notað og að sum hólf eru hugsanlega ónýtt eins og sjá má í trénu `largestTree` í `main.cpp`.

Aðalforritið (`main.cpp`), sem prófar útfærsluna ykkar er gefið og rétt úttak er í skránni `output.txt`.

English description

In this project, you need to implement the class `BinaryTree`. The interface (`binarytree.h`) is given, but you need to implement the missing parts in `binarytree.cpp`.

The nodes of the binary tree, which contains `char` data, are stored in a `vector` (but not in a linked list). By using a `vector`, each node `v` the binary tree `T` has a specific number given by the numbering function `f(v)`:

- If `v` is the root of `T`, then `f(v) = 1`
- If `v` is the left child of node `u`, then `f(v) = 2*f(u)`
- If `v` is the right child of node `u`, then `f(v) = 2*f(u) + 1`

`f(v)` thus returns an index into the underlying `vector` where the node `v` is stored. The root is, for example, stored in entry no. 1 in the underlying `vector`. Note that the first entry (no. 0) in the underlying `vector` is not used, and that some entries are possibly unused, as can be seen in the tree `largestTree` in `main.cpp`.

A main program (`main.cpp`), which tests your implementation, is given. Correct output is in the file `output.txt`.

2. BankSimulation (50%)

Lýsing á íslensku

Í þessu verkefni eigið þið að klára útfærslu á forriti sem hermir eftir þremur biðröðum í banka. Eftirfarandi skrár/klasar eru í verkefninu:

- Aðalforritið `main.cpp` er gefið.
- Klasinn `RandGenerator`. Bæði skil (`randgenerator.h`) og útfærsla (`randgenerator.cpp`) eru gefin.
- Klasinn `Bank`. Skilin `bank.h` eru gefin en þið þurfið að útfæra `bank.cpp`.

Í athugasemdum í `bank.cpp` kemur fram hvað sérhvert fall í `Bank` klasanum á að gera.

Með því að skoða dæmi um úttak, í skránni `output.txt`, sjáið þið hvernig úttak forritsins á nákvæmlega að vera.

Upplýsingar um `queue` klasann í `Standard Template Library` eru hér fyrir neðan.

English description

In this project, you need to finish the implement of a program that simulates three teller‘s queues in a bank. The following files/classes are part of the project:

- The main program `main.cpp` is given.
- The class `RandGenerator`. Both the interface (`randgenerator.h`) and the implementation (`randgenerator.cpp`) is given.
- The class `Bank`. The interface `bank.h` is given, but you need to implement `bank.cpp`.

The comments in `bank.cpp` show what each member function in the `Bank` class should do.

By inspecting the example output, `output.txt`, you see how exactly the program output should be.

Information about the `queue` class in the `Standard Template Library` are here below.

queue<T, Sequence>

Categories: containers, adaptors **Component type**: type

Description

A `queue` is an adaptor that provides a restricted subset of Container functionality. A `queue` is a "first in first out" (FIFO) data structure. That is, elements are added to the back of the `queue` and may be removed from the front; `Q.front()` is the element that was added to the `queue` least recently. `Queue` does not allow iteration through its elements.

`Queue` is a container adaptor, meaning that it is implemented on top of some underlying container type. By default that underlying type is `deque`, but a different type may be selected explicitly.

Defined in the standard header `queue`.

Template parameters

Parameter	Description	Default
`T`	The type of object stored in the queue.	
`Sequence`	The type of the underlying container used to implement the queue.	`deque<T>`

Members

Member	Description
`value_type`	The type of object stored in the `queue`. This is the same as `T` and `Sequence::value_type`.
`size_type`	An unsigned integral type. This is the same as `Sequence::size_type`.
`bool empty() const`	Returns `true` if the `queue` contains no elements, and `false` otherwise. `Q.empty()` is equivalent to `Q.size() == 0`.
`size_type size() const`	Returns the number of elements contained in the `queue`.
`value_type& front()`	Returns a mutable reference to the element at the front of the queue, that is, the element least recently inserted. Precondition: `empty()` is`false`.
`const value_type& front() const`	Returns a const reference to the element at the front of the queue, that is, the element least recently inserted. Precondition: `empty()` is`false`.
`value_type& back()`	Returns a mutable reference to the element at the back of the queue, that is, the element most recently inserted. Precondition: `empty()` is`false`.
`const value_type& back() const`	Returns a const reference to the element at the back of the queue, that is, the element most recently inserted. Precondition: `empty()` is`false`.
`void push(const value_type& x)`	Inserts `x` at the back of the queue. Postconditions: `size()` will be incremented by `1`, and `back()` will be equal to `x`.
`void pop()`	Removes the element at the front of the queue. Precondition: `empty()` is `false`. Postcondition: `size()` will be decremented by `1`.
`bool operator==(const queue&, const queue&)`	Compares two queues for equality. Two queues are equal if they contain the same number of elements and if they are equal element-by-element. This is a global function, not a member function.

`bool operator<(const queue&, const queue&)`	Lexicographical ordering of two queues. This is a global function, not a member function.