Map með tvíundarleitartré (binary search tree - BST)

Gefinn er abstract klasinn Map ásamt main forriti sem smíðar BSTMap og kallar á föllin á Map. Verkefnið er að búa til klasann *BSTMap* sem útfærir *Map* með tvíundarleitartré (*BST*). *Einnig þá hjálparklasa sem þarf.* Tréð er raðað á *lykli* sem er af taginu ***K*** en hver hnútur geymir einnig *gögn* af taginu ***T***.

Í *Map.h* er einnig lýst yfir frávikskl̈sunum *NotFoundException* og *ItemExistsException*.

BSTMap þarf að yfirskrifa öll pure virtual föllin á Map. Þau eru eftirfarandi:

- **void insert(K key, T data)**
  Bætir gögnunum *data* inn með lyklinum *key*.
  Ef þegar er stak með lykilinn *key* í gagnagrindinni er kastað ***ItemExistsException***.
- **void update(K key, T data)**
  Uppfærir gögnin tengd lyklinum *key* þannig að þau séu núna *data*.
  Ef ekkert stak með lykilinn *key* er í gagnagrindinni er kastað ***NotFoundException***.
- **T get(K key)**
  Skilar gögnunum sem tengd eru lyklinum *key*.
  Ef ekkert stak með lykilinn *key* er í gagnagrindinni er kastað ***NotFoundException***.
- **void remove(K key)**
  Fjarlægir lykilinn *key* og gögnin honum tengd úr gagnagrindinni.
  Ef ekkert stak með lykilinn *key* er í gagnagrindinni er kastað ***NotFoundException***.
- **bool contains(K key)**
  Skilar *true* ef gagnagrindin inniheldur lykilinn *key* og tengd gögn, annars *false*.
- **int size() const**
  Skilar fjölda staka sem gagnagrindin inniheldur.
- **bool empty() const**
  Skilar *true* ef gagnagrindin inniheldur engin stök, annars *false*.
- **void clear()**
  Tæmir gagnagrindina.
- **void print(ostream& out) const**
  Prentar út innihald gagnagrindarinnar.
  Þetta fall er bara fyrir ykkur að gera prófanir. Mooshak kallar ekki á það.

Verkefninu er skilað á Mooshak.

- Þið fáið ***þrjár tilraunir*** til að senda verkefnið inn í Mooshak. Þriðja tilraunin er lokaskil.
- Sendið ZIP skrá inn í Mooshak sem inniheldur allan forritstextann ykkar, .cpp og .h skrár.

Þið verðið að gera virkilega góðar prófanir í main fallinu.

- Núverandi main() forrit inniheldur hjálparföll og dæmi um notkun þeirra.
- Þið verðið að sjá til þess að jaðartilvik séu prófuð.
  - Bætið í gagnagrindina og takið úr henni í mismunandi röð.

    - Bætið í og takið úr tómri gagnagrind.
    - Athugið hvort stök eru í gagnagrindinn fyrir og eftir eyðingu.
- Verið alveg viss um að ykkar forritstexti virki og að þið skiljið hvernig og hvers vegna hann virkar áður en þið byrjið að reyna að skilja villuskilaboðin frá Mooshak.
- Verið viss um að forritið leki ekki minni.
    - Fyrir hvert new ætti að vera delete.
    - Eyðirinn ætti að eyða öllu minni sem hefur við úthlutað.

English:

You are given the abstract class Map as well as a main program that constructs a BSTMap and calls the functions defined in Map.
The assignment is to make the class *BSTMap* that implements *Map* with a Binary Search Tree (*BST*). *Also any helper classes as needed.* The tree is ordered on a *key* of the type ***K*** but each node also contains *data* of the type ***T***.

In *Map.h* you will also find definitions of the exception classes *NotFoundException* and *ItemExistsException*.

BSTMap must override all the pure virtual functions on Map. They are the following:

- **void insert(K key, T data)**
  Add the data *data* with the key *key*.
  If there is already an item with the key *key* in the map, throw ***ItemExistsException***.
- **void update(K key, T data)**
  Updates the data connected to the key *key* so they are now *data*.
  If no item with the key *key* is found in the map, throw ***NotFoundException***.
- **T get(K key)**
  Returns the data connected to the key *key*.
  If no item with the key *key* is found in the map, throw ***NotFoundException***.
- **void remove(K key)**
  Removes the key *key* and its connected data from the data structure.
  If no item with the key *key* is found in the map, throw ***NotFoundException***.
- **bool contains(K key)**
  Returns *true* the map contains the key *key* and connected data, otherwise *false*.
- **int size() const**
  Returns the number of items currently in the map.
- **bool empty() const**
  Returns *true* if the map contains no items, otherwise *false*.
- **void clear()**
  Empties the data structure.
- **void print(ostream& out) const**
  Prints the contents of the data structure.
  This function is only for your own testing. Mooshak does not call it.

The assignment will be returned through Mooshak.

- You get ***three attempts*** to enter it into Mooshak. The third one is final.
- Send a ZIP folder into Mooshak containing all your code, .cpp and .h files.

You must make really good tests in your own main function.

- The current main function contains helper functions and examples of their use.
- You must make sure that you test all edge cases.
    - Add to the data structure and remove from it in different orders
    - Add to and remove from an empty data structure
- Be absolutely sure that your code works and that you understand why it works before trying to understand Mooshak's error messages.
- Make sure your program doesn't leak memory.
    - For every new there should be a delete.
    - Deconstructor should delete everything that was allocated.