Lab 02 (8%)

Topics: ADT Set, Hash Tables, Binary Search Trees

Problem 01 (1%)

Write C++ program using standard class unordered_set which reads and executes commands of following types:

+ word (add word to set s);
- word (delete word from set s);

? word (print "YES" if set s has this word, otherwise print "NO");

(print current contexts of set s);

Word in all commands is a sequence of lowercase English letters.

Set s is a collection of unique strings. You have to use class unordered_set to represent this set in your program. Give performance characteristics of +, -, ? operations.

Use class unordered_set<string> with your own hash function which returns 42 as a hash value for any strings. Give performance characteristics in this case.

Problem 02 (3%)

Create class HashSetStr to store unique strings and use it instead of unordered_set to solve Problem 01. Class HashSetStr has to have following interface:

Constructor
HashSetStr(HashFunc hf)

Destructor
~HashSetStr()

bool insert(const string& s)
Inserts string s in hash set. Returns true if insertion was successful, otherwise returns false.

bool erase(const string& s)
Deletes string s from hash set. Returns true if deletion was successful, otherwise returns false.

bool find(const string& s) const
Searches for string s in hash set. Return true if s was found, otherwise returns false.

void clear()
Deletes all elements of container.

size_t size() const
Returns the number of elements in container.

void print() const
Prints all chains (buckets) of hash table in following order:
index of bucket: <element> <element> …

Your class has to use "Separate chaining approach" to resolve collisions of elements.
In case of command "^" you have to call method print of your hash set.

Problem 03 (4%)
(class BSTreeInt)

You have to develop class BSTreeInt which represents a collection (set) of unique integer values based on data structure called binary search tree. Test your class with unit-tests library CATCH.

Your class has to have:
constructor BSTreeInt()
Create empty ready to use set.

destructor ~BSTreeInt()
Destroy set.

method clear()
Remove all elements.

method int size() const
Return the size of the set.

method bool insert(int k)
Add k to the set; return false if set already has this number.

method bool find(int k) const
Search for k; return false if set does not have this number.

method bool erase(int k)
Remove k; return false if set does not have this number.

method void print(ostream& out) const
Print all elements to stream out in order.