Lab 05 (8%)
Topics: ADT Map, Binary Search Trees

Problem 01 (1%)

Write a program using class map from C++ standard library which reads arbitrary sequence of words from standard input and prints for each entered word the number of times it appeared. Words in output must be in increasing lexicographical order. Give performance characteristics of map's operations.

Problem 02 (1%)

Write a program using classes map and set from C++ standard library which reads arbitrary sequence of lines of text from standard input and prints for each unique word from that text all numbers of lines this word appeared in increasing order. If word appeared more than once in some line you have to print this number only once. Words must be printed in lexicographical order.

Problem 03 (1%)

Write a program using class map from C++ standard library which manages list of caches in some imaginary computer game. Program has to work with following commands:

insert <x-coordinate> <y-coordinate> <item1> <item2> …

Command adds new cache with specified coordinates and corresponding list of items.

erase <x-coordinate> <y-coordinate>

Command removes cache with specified coordinates and prints "removed".
If such a cache does not exist program prints "does not exist".

check <x-coordinate> <y-coordinate>

If such a cache exists program has to output "found" and print all items of that cache.

Your program has to use type: map<Point, vector<string>, CmpPoints> where Point is a simple structure to store x, y coordinates of some cache, vector<string> represents list of items and CmpPoints is a criterion which class map has to use to compare Points.

Problem 4 (2.5%)

Solve Problem 01 using your own class MapStrInt. Your class has to use Binary Search Tree data structure and have following methods:

MapStrInt()
Constructor: creates an empty map

~MapStrInt()
Destructor.

int& operator[](const std::string& k):
Returns reference to integer which associates with the key k.

void printInOrder() const;
Prints the content of the map in the format
size <size>: { (k1, v1) (k2, v2) … }
Strings k1, k2, … must be in increasing lexicographical order.

void clear();
Removes all elements of the map.

int size() const;
Returns the number of elements in the map.

Problem 04 (2.5%)

Add following methods to the class MapStrInt:

std::pair<bool, int> find(const std::string& k) const;
Searches for a key k and returns a pair where the first field is a signal of success, second field is a value associated with the key k (if search is unsuccessful it is 0)

bool insert(const std::string& k, int v);.

bool erase(const std::string& k);

Write simple program to test these methods. This program has to work with following commands to manage the object of class MapStrInt:

print
    Command prints the content of the map.
find <key>
    if <key> is in map program prints “found” and the value associated with that <key>.
    Otherwise it has to print “not found”
update <key> <value>
    if <key> in the map program has to change its value to <value>.
    Otherwise it has to add new pair (<key>, <value>) to the map.
insert <key> <value>
    If map has already this key program has to print “not inserted”.
    Otherwise it has to add new pair (<key>, <value>) to the map and print “inserted”.
erase <key>
    If map does not have this key program has to print “not erased”.
    Otherwise it has to erase the pair with this key and print “erased”.