CSE 330 LABORATORY -- Week 9, Winter 2020

Instructor: Kerstin Voigt

This lab will have you work with the **Map ADT**. A simple application is given in form of a "wager game" in file MapGame.cpp. As we have discussed in the lecture, we base our Map data structure implementation on the Set ADT.

You will be using the following files as they are:

1. Vector.h
2. Stack.h
3. MapMain.cpp

You will need to modify Set.h to match up properly with the needs of a new file Map.h. Make a copy of Set.h and call it SetM.h

Then make the following modifications and additions to SetM.h:

1. The *-operator of the Set iterator must be return a refence argument (despite that not being desirable for the generic Set data type. Code should look like this:

```
class iterator
{
   <cut>

  C& operator *()
  {
     return current->element;
  }

  <cut>
};
```

2. The insert member functin of the Set data type must return an iterator to the inserted item. Code should be this:

   Under public:

```
//void insert( const C & x )

iterator insert(const C & x)
{
   Stack<Vector<BinaryNode*> > st;
   BinaryNode* p = insert( x, root, st );
   return iterator(p, st);
}
```

Under private:

```
BinaryNode* insert( const C& x, BinaryNode * & t,
                            Stack<Vector<BinaryNode*> > & st)
{
   if (t == nullptr)
   {
       t = new BinaryNode{ x, nullptr, nullptr };
       return t;
   }
   else if (x < t->element)
   {
       st.push(t);
       return insert(x, t->left, st);
   }
   else if (t->element < x)
   {
       st.push(t);
       return insert(x, t->right, st);
   }
   else
       return t;
}
```

Your will produce

1. Your own file Map.h according the the lecture notes. (Extended version.)
2. Your own file Pair.h as show below.

Exercise 1: Put the above pieces together. Test and debug until you have a running game program. Play a few rounds with some classmates – it is multi-player.

You may like this game a lot …. or not at all. Either way, it can certainly use some improvement. For this proceed to Exercise 2.

Exercise 2: Decide on the types of improvements to the "game engine" (not the user interface) that could make this game more interesting or in any way more enjoyble to play. Make those improvements. Test, debug, test some more, and demonstrate.

Credit for this lab: (1) Sign up on the signup sheet. (2) Submit via portal, your files Pair.h, SetM.h, and Map.h and a typescript/ screen shot of a *4-round game between at least three players*. The submission of your *working* Lab9 program will be worth 5 extra homework points (note: 5 or zero).

A submission portal will be kept open until March 16, 11:59pm.

Code for Lab9 see next page

```
// Pair.h

#ifndef PAIR_H_
#define PAIR_H_
using namespace std;

template <typename K, typename V>
class Pair
{
 public:

  Pair() {}

  Pair(K thekey)
    :first(thekey) {}
  Pair(K thekey, V theval)
    : first(thekey), second(theval) {}

  Pair(const Pair& rhs)
    : first(rhs.first), second(rhs.second) {}

  bool operator == (const Pair<K,V>& rhs) const
  {
    return  first == rhs.first;
  }

  bool operator != (const Pair<K,V>& rhs) const
  {
    return first != rhs.first;
  }

  bool operator < (const Pair<K,V>& rhs) const
  {
    return first < rhs.first;
  }

  bool operator > (const Pair<K,V>& rhs) const
  {
    return first > rhs.first;
  }

  K first;
  V second;
};

#endif
```