```
//================================================================
// Hash.cpp
// This file contains exceprts from the Hash Table class and
// Dictionary class.
//================================================================

//================================================================
// HASH TABLE
//================================================================
template <class KeyType>
class HashTable
{
public:
                                                HashTable       (int numSlots);
                                                HashTable       (const HashTable<KeyType>& h);
                                                ~HashTable      ();
        KeyType*                        get                     (const KeyType& k) const;
        void                            insert          (KeyType *k);
        void                            remove          (const KeyType& k);
        HashTable<KeyType>& operator=   (const HashTable<KeyType>& h);
        std::string toString                            (int slot) const;

private:
        int                     slots;
        List<KeyType>   *table; // an array of List<KeyType>â\200\231s
};

//================================================================
// default constructor (unspecified slots)
//================================================================
template <class KeyType>
    HashTable<KeyType>::HashTable(void)
{
  slots = 10;
  table = new List<KeyType>[slots];
}

//================================================================
// default constructor (specified slots)
//================================================================
template <class KeyType>
    HashTable<KeyType>::HashTable(int numSlots)
{
    slots = numSlots;
    table = new List<KeyType>[slots];
}
//================================================================
// get
// this method returns a pointer to the object in the hash
// table where the value resides
// parameters: const KeyType& k for which to find
// return value: KeyType* location
//================================================================
template <class KeyType>
KeyType* HashTable<KeyType>::get(const KeyType& k) const
{
  int index = k.hash(slots);
  return table[index].get(k);
}

//================================================================
// insert
// this method inserts a value into the hash table
// parameters: KeyType* k to insert
// return value: void
//================================================================
template <class KeyType>
void HashTable<KeyType>::insert(KeyType* k)
{
  int slot = k->hash(slots);
  table[slot].insert(0,k);
}

//================================================================
// remove
// this method removes a given value from the hash table
// parameters: const KeyType& k to remove
// return value: void
//================================================================
template <class KeyType>
void HashTable<KeyType>::remove(const KeyType& k)
{
  int slot = k.hash(slots);
  table[slot].remove(k);
}
//================================================================
// DICTIONARY
//================================================================
template <class KeyType>
class Dictionary: public HashTable<KeyType>
{
/*
public:

              Dictionary  (){};          //default constructor
              ~Dictionary (void){};    //destructor
              Dictionary  (const Dictionary<KeyType> &d){};
  void        insert      (KeyType *k)
  {
    HashTable<KeyType>::insert(k);
  };
  void        remove      (const KeyType &k)
  {
    HashTable<KeyType>::remove(k);
  };
  KeyType *   get         (const KeyType &k) const
  {
    HashTable<KeyType>::get(k);
  };
  bool        Empty       (void)  const
  {
    HashTable<KeyType>::Empty();
  };

private:
  HashTable<KeyType> *d;
*/

public:

        Dictionary():HashTable<KeyType>(){};
        ~Dictionary(void) {};
Dictionary(const Dictionary<KeyType> &d):HashTable<KeyType>(d){};

//Inherited HashTable Class incorrectly, much simpler this way.
};
```