

CSCI 235 - Spring 2015

Homework 1

Due: Tuesday, February 24th, 2015, 11:59 pm

Follow the instructions presented in the **Programming Rules** document (in the Course Information folder). Submit only the header and source files as well as your Makefile (I will cover this in class). Note that your program needs to compile in order to get any credit at all.

Adhere to the style guidelines. Start Early!

1 The Set Abstract Data Type

1.1 Part I

Provide a *complete templated* array based implementation of the Set Abstract Data Type. The Set ADT is the same as the Bag ADT with the only difference that duplicate items are not allowed. The items do not have to be in sorted order.

You can (and should) copy the code provided for the implementation of the Bag ADT. The only function that needs modification is `Add()` and the function `GetFrequencyOf()` is not needed.

Additionally, add a constructor that constructs a Set for a single item:

```
Set(const ItemType &an_item);
```

Similarly to the Bag implementation provide four files: **SetInterface.h**, **Set.h**, **Set.cpp** and **TestSet.cpp**.

In the file `TestSet.cpp` provide a number of test cases that demonstrate that your implementation is correct. Do this by writing a function named `TestSetImplementation()` that contains code that does the following:

1. Ensures that an empty set contains no items. This can be done as follows:

```
Set<int> a_set;
cout << "This is the empty set, and IsEmpty() returns  "
      << a_set.IsEmpty() << endl;
```

```
// Use the above logic for the all the following tests.
```

2. Adds one item to an empty set, and then searches for it (it should be there).
3. Creates a set by adding the following items: 1, 10, 3, 10, 5, 10 in that order.
4. Ensures that the set now holds only 4 items.
5. Ensures that the search (function `Contains()`) of an item in the Set returns true, and that the search of an item not in the Set returns false.
6. Ensures that trying to add more items than the maximum capacity, results in a false flag returned by `Add()`.
7. Ensures that adding an item already in the Set results in a false flag returned by `Add()`.
8. Ensures that removing an item from an empty set results in a false flag returned by the function `Remove()`.
9. Ensures that you can remove a given item from the Set.

1.2 Part II

Write two templated client functions (i.e. functions that are not part of the Set class; you can place them at the end of the Set.cpp file, but after the class implementation):

1. A function named `DisplaySet()` that gets as an argument a set, and displays the contents of the set. Here is the signature of the function:

```
template <class ItemType>
void DisplaySet(const Set<ItemType> &a_set) {
    ...
}
```

2. A function named `UniteSets(...)` that gets as arguments two Sets `set1` and `set2`. The function `UniteSets()` creates a new set that contains all elements in sets `set1` and `set2`. This new set is returned by the function. Note that sets `set1` and `set2` shouldn't be modified. Here is the signature of the function:

```
template <class ItemType>
Set<ItemType> UniteSets(const Set<ItemType> &set1,
                       const Set<ItemType> &set2) {
    ...
}
```

In the file `TestSet.cpp` provide a number of test cases for the `UniteSets()` function. Do this by writing a function `TestUniteSets()` that contains code that does the following:

- (a) Ensures that the union of two empty sets is an empty set.
- (b) Ensures that if one of the two input sets is empty, the result equals the nonempty set. Use the function `DisplaySet()` to print the sets (same holds for the next tests).
- (c) Use two sets that do not have common elements (for instance $\{1, 20, 30\}$ and $\{40, 50\}$) and ensure that the result is correct.
- (d) Use two sets with common elements (for instance $\{1, 20, 30\}$ and $\{20, 30, 0, 40\}$) and ensure that the result is correct.

2 Submission

Once you have ensured that your program works on the Linux machines in Lab 1000G in Hunter North, pack all files in a zip/tar.gz archive and name it `<YOUR_FULL_NAME>_hw1.zip/tar.gz`. Replace `<YOUR_FULL_NAME>` with your full name, using underscores for spaces.

Submit this file on Blackboard in the section for Homework 1.

Please make sure you have included the Makefile and that your program compiles using the `make all` command. Your program will not be graded at all if any of the following is true:

- The files were not archived (zipped)
- There is no Makefile
- There is a compile error
- It's missing any of the files required by this assignment

Again, please read the Programming Rules document on Blackboard.

Good luck!