A Different Setting

> This is a **pair** exercise and must be competed in your **tutorial** or **lab** with your partner.

Here's another ADT: the Set ADT. The Set ADT, like the Stack and Queue ADTs, model a concept you're likely to encounter elsewhere; a set is a cool mathematical construct, which stores a collection of unique values of the same type.

Download `Set.h`, or copy it into your current directory on a CSE system by running

```
$ cp /web/cs1511/17s2/week12/files/Set.h .
```

The Set ADT defines these methods in its interface:

- ```
  Set newSet (void);
  ```

  Create a new `Set`.

- ```
  void destroySet (Set);
  ```

  Release all resources associated with a `Set`.

- ```
  void setAdd (Set, item);
  ```

  Add an `item` to the `Set`. If the `item` already exists in the set, it does nothing.

- ```
  void setRemove (Set, item);
  ```

  Remove an `item` from the `Set`. If the `item` does not exist in the set, it does nothing.

- ```
  bool setContains (Set, item);
  ```
  ```

Does the `Set` contain this `item`? Returns `true` or `false`.

- ```
  Set setUnion (Set, Set);
  ```

  Take the union of two sets ($a \cup b$), and return the resulting set. The union of two sets is the set containing all the unique `item`s of both sets.

- ```
  Set setIntersection (Set a, Set b);
  ```

  Take the intersection of two sets ($a \cap b$), and return the resulting set. The intersection of two sets is the set containing all the `item`s that are common to both sets.

- ```
  bool setSubset (Set a, Set b);
  ```

  Is `a` a subset of `b` ($a \subseteq b$)? That is, does `a` contain all the `item`s that `b` cotains?

- ```
  bool setEqual (Set a, Set b);
  ```

  Returns `true` if `a` is equal to `b`, or `false` otherwise. If $a \subseteq b$ and $b \subseteq a$, then $a \equiv b$; Or, to put it another way, if `a` contains all the `item`s in `b`, and `b` contains all the `item`s in `a`, `a` and `b` are equal.

Create a file called `Set.c`; in it, you should implement these functions. You probably should use a linked list to store the different `item`s.

To run some simple automated tests:

```
$ 1511 autotest setADT
```

To run Styl-o-matic:

```
$ 1511 stylomatic Set.c
Looks good!
```

You'll get advice if you need to make changes to your code.

Submit your work with the *give* command, like so:

```
$ give cs1511 wk12_setADT
```

Or, if you are working from home, upload the relevant file(s) to the `wk12_setADT` activity on Give Online.