9. Sets

This assignment is about Abstract Data Types (ADTs). One such ADT, to implement a simple (resizeable) 1D Array, is provided and then you are expected to create one, in a similar style, to implement sets. The framework for testing both the Array ADT, and your own Set ADT (when you've finished it) are provided in the form of test `.c` files, and a `Makefile` to take care of the compilation.

On the web page are the files `arr.h`, `arr.c`, `testarr.c` and `Makefile`. These files enable you to build, and test, the ADT for a 1D Indexed Array. This simple replacement for C arrays is 'safe' in the sense that if you write the array out-of-bounds, it will be automatically resized correctly (using `realloc()`). The interface to this ADT is in `arr.h` and its implementation is in `arr.c`. Typing `make testarr`, compiles these files together with the test file `testarr.c`. Executing `./testarr` should result in all tests passing correctly.

```
% make testarr
% ./week9p2
Basic Set Tests ... Start
Basic Set Tests ... Stop
```

Notice the underlying type of the array (`arrtype`) is fixed to be an `int` by `#define ARRINTS`. In this case, it is done inside the `Makefile` itself using a -D flag. There's also a way of setting the underlying type to be string (inside a structure), but you don't need to worry about this until the final part.

Exercise 9.1 Build `testarr`, and check that you understand the use of the functions, including initialization, reading, writing and freeing. ■

0%

Sets are an important concept in Computer Science. They enable the storage of elements (members), guaranteeing that no element appears more than once. Operations on sets include initializing them, copying them, inserting an element, returning their size (cardinality), finding if they contain a particular element, removing an element if it exists, and removing one element from a random position (since sets have no particular ordering, this could be the first element). Other set operations include union (combining two sets to include all elements), and intersection (the set containing elements common to both sets).

www https://www.mathsisfun.com/sets/sets-introduction.html

www https://en.wikipedia.org/wiki/Set_(mathematics)

The definition of a Set ADT is given in `set.h`, and a file to test it is given in `testset.c`.

Exercise 9.2 Write `set.c`, so that:

```
% make week9p2
cc -O3 -D ARRINTS testset.c arr.c set.c -o week9p2 ...etc...
% ./week9p2
Basic Set Tests ... Start
Basic Set Tests ... Stop
```

works correctly. Your Set ADT will build on top of the Indexed Array ADT introduced above. Only submit `set.c`. Alter no other files, including `arr.c`, `arr.h`, `set.h` or `Makefile`.
■

80%

On the web page are two text files, each containing a Jane Austin novel (cleaned up, one word per line). The words that are common to both novels could be found by creating two sets, each of which having the words from one of the files, and then computing the intersection of these.

Exercise 9.3 Write the program `janeaustin.c` to display the number of words in this intersection. Check that the existing `Makefile` can successfully build and execute `week9p3`. You will use the Set ADT created above, but it will now use the underlying string type engaged by using `-D ARRSTRINGS` in the `Makefile`. The only output will look like :

```
There are xxxx unique words in sense-and-sense.txt
There are xxxx unique words in pride-and-prej.txt
There are xxxx common words
```

but here, the exact numbers have been obfuscated. I'll test your code by checking these numbers (possibly on other files) automatically, so make sure the output is in this form exactly.
■

20%

Hints

- Make sure your final submission consists **only** of `set.c` and `janeaustin.c`. I'll use the files provided (and other test files) to check your code. Some of this will be automatic, so adhering to the naming guidelines etc. is important.
- If you can't get all the ADT implemented, comment your code at the top, explaining which functions work, and which don't. In this case, provide your own test file `mytestset.c`, which demonstrates the successful functionality of your (partial) solution.
- The quality of your code is being assessed. I'll read it in detail, and also run more tests, other than those provided.