USC Viterbi
School of Engineering

CSCI 104
Abstract Data Types

Mark Redekopp
David Kempe

USC

XKCD #138

Courtesy of Randall Munroe @ http://xkcd.com

Abstract Data Types

- DAPS defines an **abstract data type**, or ADT, as:
  - Specification/model for a group of values/data and the operations on those values
- The model allows us to separate…
  - The decision of what data structure to use and how it will be used in our higher level application
  - And the implementation of the specific data structure
- DAPS defines a **data structure** as:
  - An implementation of an ADT in a given programming language
- Each ADT we will examine in this course has certain:
  - Well defined operations and capabilities that are often useful
  - Time & space advantages
  - Time & space disadvantages
- You need to know those operations, advantages and disadvantages

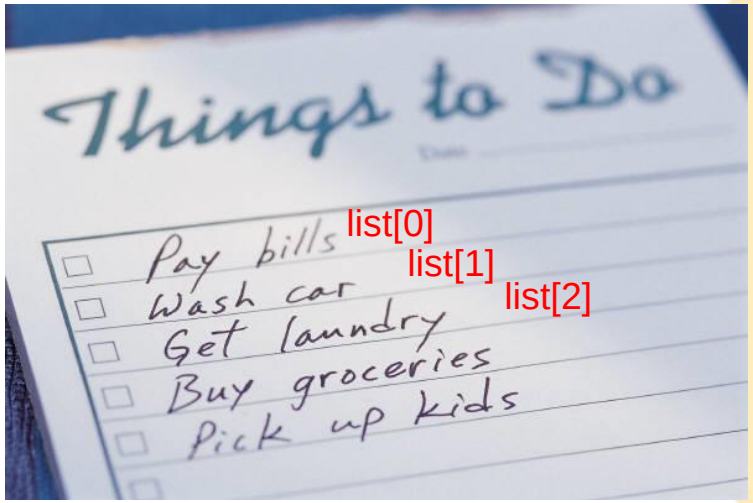
Data Abstraction & Problem Solving with C++, Carrano and Henry will henceforth be abbreviated as DAPS

3 Popular ADTs

- List
- Dictionary/Map
- Set

Lists

- Ordered collection of items, which may contain duplicate values, usually accessed based on their position (index)
  - Ordered = Each item has an index and there is a front and back (start and end)
  - Duplicates allowed (i.e. in a list of integers, the value 0 could appear multiple times)
  - Accessed based on their position (list[0], list[1], etc.)
- What are some operations you perform on a list?



USC Viterbi School of Engineering

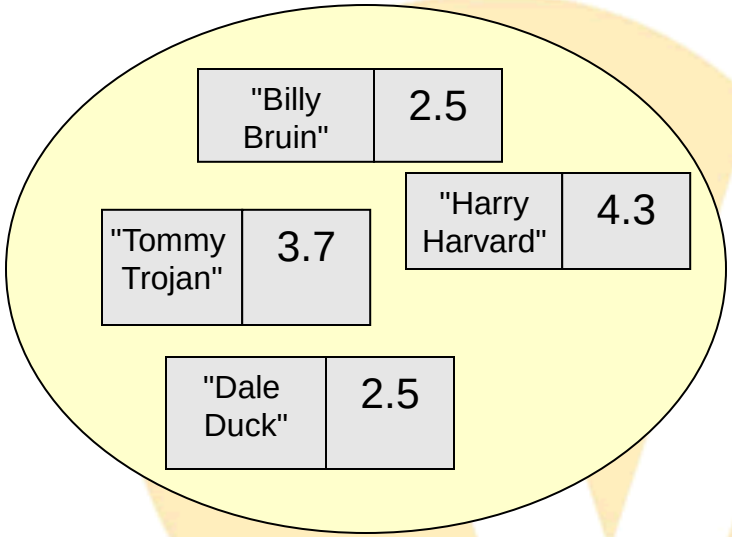
List Operations

Operation	Description	Input(s)	Output(s)
push_back / append	Add a new value to the end of the list	Value	
insert	Add a new value at a particular location shifting others back	Index : int Value	
pop	Remove value at the given location	Index : int	Value at location
at / get	Get value at given location	Index : int	Value at location
empty	Returns true if there are no values in the list		bool
size	Returns the number of values in the list		int
remove	Remove a value	Value	bool : true if removed successfully
find	Return the location of a given value	Value	Int : Index

USC

Maps / Dictionaries

- Stores key,value pairs
  - Example: Map student names to their GPA
- Keys must be unique (can only occur once in the structure)
- No constraints on the values
- What operations do you do on a map/dictionary?
- No inherent ordering between key,value pairs
  - Can't ask for the 0th item…



USC Viterbi School of Engineering

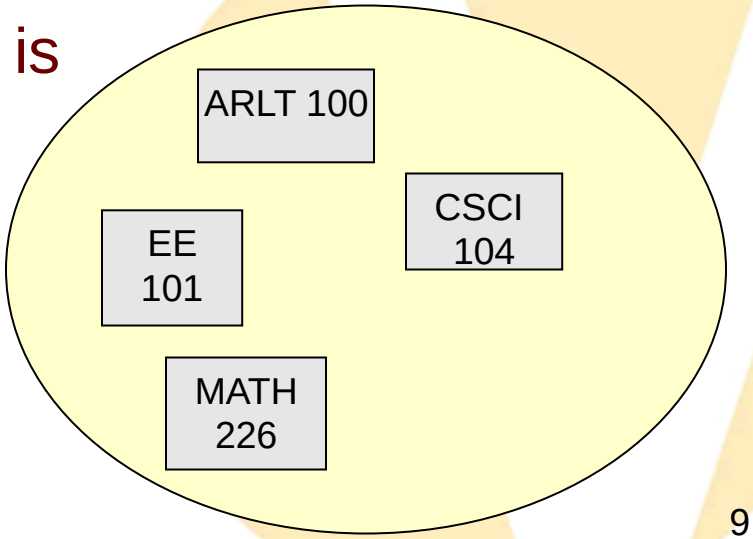
Map / Dictionary Operations

Operation	Description	Input(s)	Output(s)
Insert / add	Add a new key,value pair to the dictionary (assuming its not there already)	Key, Value	
Remove	Remove the key,value pair with the given key	Key	
Lookup / Get	Lookup the value associated with the given key or indicate the key,value pair doesn't exist	Key	Value associated with the key
In / Find	Check if the given key is present in the map	Key	Int: value at location
empty	Returns true if there are no values in the list		bool
size	Returns the number of values in the list		int

USC

Set

- A set is a dictionary where we only store keys (no associated values)
  - Example: All the courses taught at USC (ARLT 100, …, CSCI 104, MATH 226, …)
- Items (a.k.a. Keys) must be unique
  - No duplicate keys (only one occurrence)
- Not accessed based on index but on value
  - We wouldn't say, "What is the 0^{th} course at USC?"
- In DAPS textbook Chapter 1, this is the 'bag' ADT
- What operations do we perform on a set?



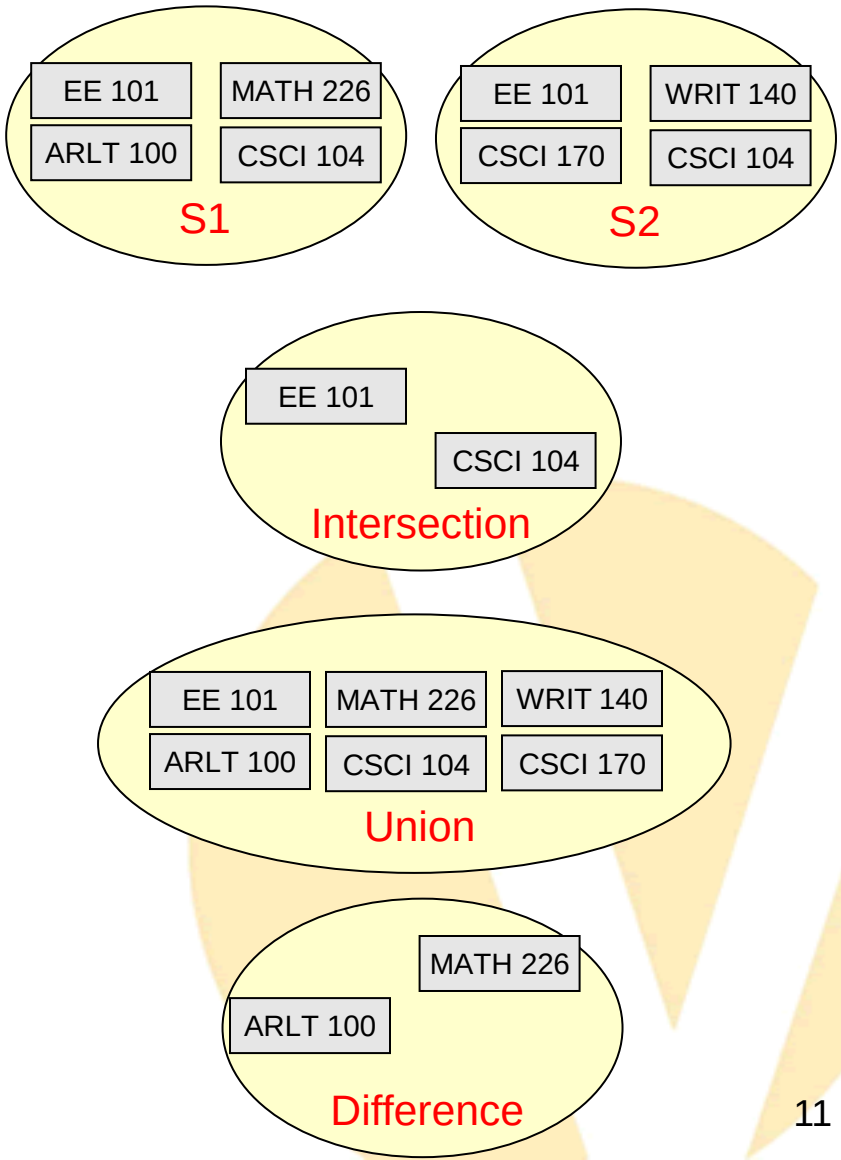
USC Viterbi School of Engineering

Set Operations

Operation	Description	Input(s)	Output(s)
Insert / add	Add a new key to the set (assuming its not there already)	Key	
Remove	Remove	Key	
In / Find	Check if the given key is present in the map	Key	Int: value at location
empty	Returns true if there are no values in the list		bool
size	Returns the number of values in the list		Int
intersection	Returns a new set with the common elements of the two input sets	Set1, Set2	New set with all elements that appear in both set1 and set2
union	Returns a new set with all the items that appear in either set	Set1, Set2	New set with all elements that appear in either set1 and set2
difference	Returns a set with all items that are just in set1 but not set2	Set1, Set2	New set with only the items in set1 that are not in set2

Intersection, Union, Difference

- May be familiar from CS 170
- Set intersection
  - S1 ∩ S2
- Set Union
  - S1 ∪ S2
- Set Difference
  - S1 – S2

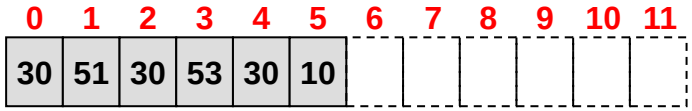


What's Your ADT?

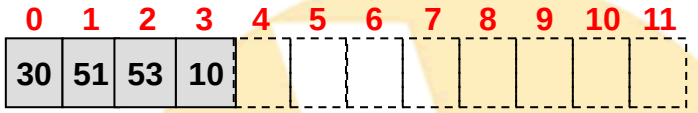
Scores on a test	List
Students in a class	Set (maybe List)
Courses & their enrollment	Map (Key = course, Value = enrollment)
Temperature Reading at a location	List
Usernames and password	Map
Index in a textbook	Map
Facebook friends	Set
Adjacent countries of a map	Set

Some Implementation Details

- List
  - An array acts as a list
  - Index provides ordering
    - First at location 0
    - Last at location n-1



- Set
  - Can use an array
  - Must check for duplicate on insertion
    - O(n) solution
  - Can we do better? Yes…



- Map
  - Can also use an array
  - Again check for duplicate key on insertion

```
struct Pair{
  string key;
  double value;;
};
```