Hashes, unions, sets

- Hashes, or associative arrays, act like a lookup table – associating values with a set of unique keys, and are implemented as a built in part of some languages
- Languages may restrict the types of values that can be used for keys and/or as values
- Typical operations are adding/modifying/removinig key value pairs, listing or iterating through keys or values
- Implementations use typical searchable data types (btrees, hash tables, etc) indexed on keys and possibly with secondary index on values

unions

- For statically typed languages, unions provide a means of grouping together data types, e.g. we might create a weekday type in which values may be either a string (e.g. “Friday” or an integer value corresponding to that day (e.g. 5)
- They often use a structure like syntax to associate value or field names with data types, e.g.

```
Union Weekday { string DayAsStr; int DayAsInt; }

WeekDay D;

D.DayAsInt = 6;
```

- Allocated storage must be large enough for the largest storable type
- Automated type checking may or may not be supported, e.g. if we assign 6 to D.DayAsInt and then try to access D.DayAsStr, what happens?

Sets

- Some languages will provide sets as built in types, others may provide them through libraries or modules
- Implementation may be as a bit vector (in/not-in a set of known values) or as a form of list (all items in the list are part of the set)
- Typical set operations are generally supported (union, intersection, complement (for bit-vector representations), etc.