A Generic Set ADT

Project 4

Generic Set ADT

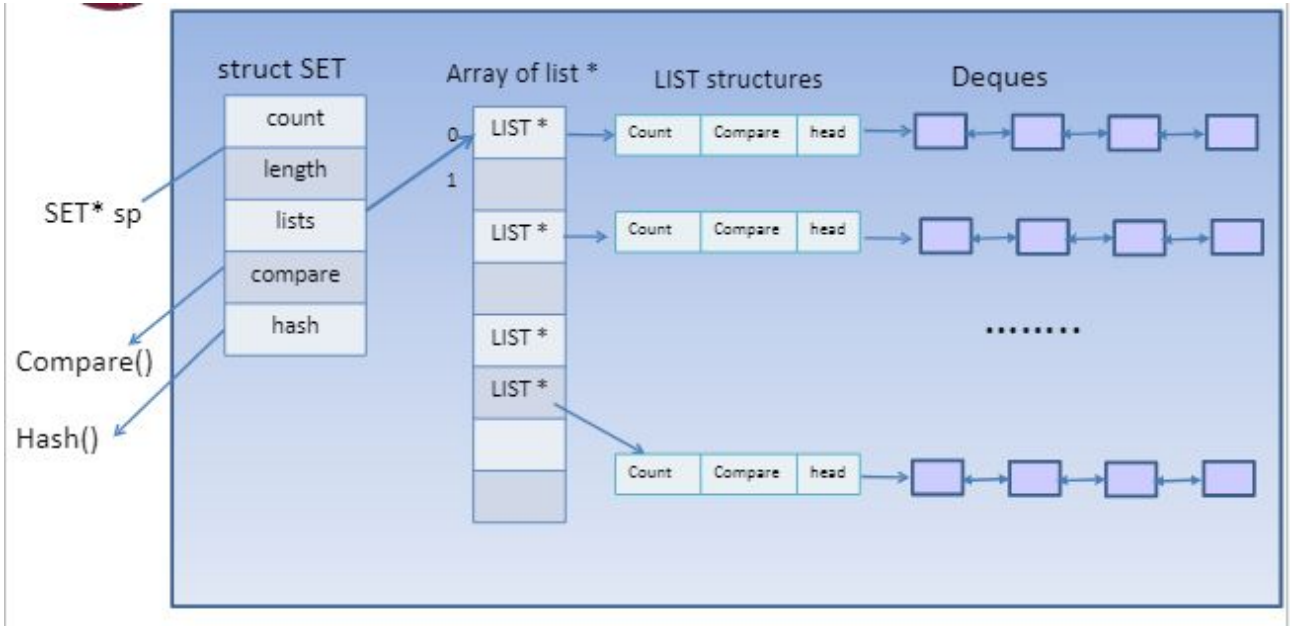
- set.c
- Create set.c to implement the set operations with generic data type

```
struct set {
        int count;
        int length;
        LIST **lists;
        int (*compare)();
        unsigned (*hash)();
        };
```

```
SET *createSet(int maxElts, int (*compare)(), unsigned (*hash)());
void destorySet(SET *sp);
int numElements(SET *sp);
void addElement(SET *sp, void *elt);
void removeElement(SET *sp, void *elt);
void *findElement(SET *sp, void *elt);
void *getElements(SET *sp);

(*sp->compare) (elt1, elt2)
(*sp->hash)(elt)
```

Data Structure



- lists
  - The hashtable
  - Each index points to a list
  - Use the list ADT from week 1!
  - Use chaining in collisions

Generic Set Helpful Tips

- Use functions from list ADT (list.h)!

- void *getElements(SET *sp)
  - Allocates and returns **one** array of the elements in **all** the lists in set
  - Can use getItems() + memcpy()
    - memcpy() will work here because getItems() returns **consecutive** memory

- void destroySet(SET *sp)
  - Make sure to free lists
    - Use destroyList()

Submission

- **Document Big-O for all functions**
- Submission deadline:
  - Sunday, May 16th at 11:59 pm
  - 10% off every 24 hrs after deadline
  - No submission will be accepted after Wed, May 19th at 11:59 pm
- Demo deadline:
  - Lab section next week
  - No demo will be accepted in TA's office hours after your lab section next week
- File Submission:
  - Both **tar file or zip file** will be accepted
    - tar -czvf project3.tar folder_path
    - folder_path is the directory of the folder that contains both set.c and list.c
  - Can also use mksolution