

Computer Engineering 12

Project 3: Sets, Arrays, and Hash Tables

Due: Sunday, 10/26 at 11:59 pm

Introduction

In this project, you will implement a **set** abstract data type for strings. Your interface and implementation must be kept separate. Separate source files that provide main will be provided for testing your data type.

Interface

The interface to your abstract data type must provide the following operations:

- `SET *createSet(int maxElts);`
//return a pointer to a new set with a maximum capacity of *maxElts*
- `void destroySet(SET *sp);`
//deallocate memory associated with the set pointed to by *sp*
- `int numElements(SET *sp);`
//return the number of elements in the set pointed to by *sp*
- `bool hasElement(SET *sp, char *elt);`
//return whether *elt* is a member of the set pointed to by *sp*
- `bool addElement(SET *sp, char *elt);`
//add *elt* to the set pointed to by *sp*, and return whether the set changed
- `bool removeElement(SET *sp, char *elt);`
//remove *elt* from the set pointed to by *sp*, and return whether the set changed

Implementation

Implement a set using a hash table of length $m > 0$ and linear probing to resolve collisions. Create an auxiliary function `findElement` that contains all of the search logic as you did for the previous assignment and use `findElement` to implement the functions in your interface. The following hash function should be used:

```
unsigned hashString(char *s) {  
    unsigned hash = 0;  
    while (*s != '\0')  
        hash = 31 * hash + *s++;  
    return hash;  
}
```

Submission

Create a directory called `project3` to hold your solution. Call the header file containing your interface `set.h` and the source file `table.c`. Create a file called `report.txt` containing the results requested for below. Submit a zip (or tar) file containing the `project3` directory using the online submission system of `camino`.

Grading

Your implementation will be graded in terms of correctness, clarity of implementation, and commenting and style.

Your implementation **must** compile and run on the workstations in the lab. The algorithmic complexity of each function **must** be documented. Report the execution times of the test programs on each of the sample input files by using the time command. (Report the average of the “real” times of at least three runs on each input file.)