15-122: Principles of Imperative Computation — Spring 2020

Lab 9: Legacy of the void* — Tuesday March 17th

Collaboration: In lab, we encourage collaboration and discussion as you work through the problems. These activities, like recitation, are meant to get you to review what we've learned, look at problems from a different perspective and allow you to ask questions about topics you don't understand. We encourage discussing problems with your neighbors as you work through this lab!

Setup: Copy the lab code from our public directory to your private directory:

```
% cd private/15122
% mkdir lab09
% cd lab09
% wget https://web2.qatar.cmu.edu/~srazak/courses/15122-s20/lab/handout-09.tgz
% tar xfvz handout-09.tgz
```

Grading: Finish tasks (1.a) to (1.c) for full credit, and additionally finish (1.d) for extra credit.

Using generic hash tables

In this lab, we'll be using the hash dictionaries discussed in lecture, but we'll be implementing a slightly different dictionary interface than what we saw in class.

```
/*** Client interface ***/

typedef void* key;
typedef void* value;

typedef bool key_equiv_fn(key x, key y);
typedef int key_hash_fn(key x);

/*** Library interface ***/

// typedef ______* hdict_t;
typedef struct hdict_header* hdict_t;

hdict_t hdict_new(int capacity, key_equiv_fn* equiv, key_hash_fn* hash)
  /*@requires capacity > 0 && equiv != NULL && hash != NULL; @*/
  /*@ensures \result != NULL; @*/ ;

value hdict_lookup(hdict_t H, key k)
  /*@requires H != NULL; @*/ ;

void hdict_insert(hdict_t H, key k, value v)
  /*@requires H != NULL && v != NULL; @*/
  /*@ensures hdict_lookup(H, k) == v; @*/ ;
```

Our sample application will be used in checking student attendance. Your code for this should go in a file called `rollcall.c1`.

(1.a) Define a struct that represents students. Its fields should include `andrew_id` (**`string`**),

`days_present` (**`int`**), and `days_absent` (**`int`**). You can include other fields if you want, but you need these fields with these types.

Write out the definition of this struct. Include a **`typedef`** so that you can allocate structs with **`alloc`**`(student)`.

1.5pt

(1.b) Write client functions for a hashtable based on student information. For this lab we will think of our keys as being Andrew IDs, and therefore be using pointers to **`string`**s (**`string`**$*$) to represent them. We will think of the entries as being students, and therefore use pointers to `student`s (`student`$*$) to represent the value.

Hint: Your functions should have the requirement that `x` and `y` are both non-`NULL` and have **`string`**$*$ as their tag.

```
int hash_student(key x);
bool students_same_andrewid(key x, key y);
```

(1.c) Write a function that initializes a `hdict_t` with students that have no attendance record. Don't worry about what happens if there are duplicates in this array.

```
hdict_t new_roster(string[] andrew_ids, int len)
//@requires \length(andrew_ids) == len;
```

3pt

At this point, you should create a trivial `main()` function inside `rollcall.c1` just to make sure your code compiles:

```
cc0 -d hdict.c1 rollcall.c1
```

You'll need to delete this `main()` function before compiling with `test-rollcall.c1` below.

(1.d) Write functions that increment a student's attendance record.

```
void mark_present(hdict_t H, string andrew_id)
//@requires H != NULL;

void mark_absent(hdict_t H, string andrew_id)
//@requires H != NULL;
```

These functions should manipulate the `days_present` and `days_absent` fields stored in the hash table, so that `hdict_lookup` can access these fields later on.

4pt

You can compile and run your code with `test-rollcall.c1`:

```
% cc0 -d hdict.c1 rollcall.c1 test-rollcall.c1
% ./a.out
Enrolling bovik, rjsimmon, fp, and niveditc... done.
Student gburdell is not enrolled...
Student bovik is enrolled...
Student rjsimmon is enrolled...
Student twm is not enrolled...

Student bovik: 5 present, 4 absent...
Student rjsimmon: 8 present, 1 absent...
Student niveditc: 8 present, 1 absent...
Student fp: 2 present, 7 absent...
Done!
```