

Class 31: Debrief on Midsemester Examination

Held: Monday, 18 April 2011

Summary: We consider a few issues from the mid-semester examination.

Related Pages:

- EBoard.
- Reading: Graham, Paul (2002). *Revenge of the Nerds*. Graham, Paul (2002). *What Made Lisp Different*.

Notes:

- Readings for Wednesday: McCarthy, J. (1978). History of LISP. *ACM SIGPLAN Notices* 13(8), pp. 217-223. DOI=<http://doi.acm.org/10.1145/960118.808387>.
- Exam grading delayed. Expect exams back on Friday.
- Due to prereg and College committee work, I need folks to sign up for appointments. Times at <http://www.cs.grinnell.edu/~rebelsky/Schedule/signup.2011S.txt>. Signup via email.
- Retrospective EC for Drag show.
- EC for Ginsburg convo, Thursday.
- EC for open discussion with Jane Ginsburg, Thursday, 9:00-10:30 in Mears Cottage.
- EC for open discussion with Jane Ginsburg, Thursday, 2:30-4:00 in Mears Cottage.

Overview:

- Hash tables.
- General Issues.

Implementing Hashes

- Sam's code `Examples/MidSem/HashTable`.
- A (somewhat) nice idea from a few students: Implement them as tries.
- Some sample sources:
 - <http://www.sparknotes.com/cs/searching/hashtables/section3.rhtml>. Buggy.
 - http://en.literateprograms.org/Hash_table_%28C%29. Better, but only if you read the whole thing. Bad testing.

Problematic code. See if you can figure out why I'm unhappy with it.

Example 0:

```

int dict_add(dict_t *hashtable, char *key, char *value)
{
    list_t *new_list;
    char *current_key;
    unsigned int hashval = hash(hashtable, key);

    /* Attempt to allocate memory for list */
    if ((new_list = malloc(sizeof(list_t))) == NULL) return 1;

    /* Does item already exist? */
    current_key = dict_get(hashtable, key);
    /* item already exists, don't insert it again. */
    if (current_key != NULL) return 1;
    /* Insert into list */
    new_list->key = strdup(key);
    new_list->value = strdup(value);
    new_list->next = hashtable->table[hashval];
    hashtable->table[hashval] = new_list;

    return 0;
}

```

Example 1:

```

/*
 * Add the key/value pair to the dictionary d. If that key is already
 * assigned a value, replace that value with this function's value
 * parameter. This function should run in amortized constant time. The
 * return value is 1 on success and 0 when memory allocation fails.
 *
 * Preconditions:
 *   d has been created with dict_new
 *   key and value are terminated with a null byte
 *
 * Postconditions:
 *   d will contain the key/value pair
 *   key and value will not be changed, but d will contain those actual
 *   strings, not copies of them
 */
int dict_add(dict d, char *key, char *value)
{
    bucket_t *current_chain_link = d->buckets + hash_function(key);

    /*
     * If the bucket is empty, add the key and value.
     */
    if (current_chain_link->key == NULL) {
        current_chain_link->key = key;
        current_chain_link->value = strdup(value);
        if (current_chain_link->value == NULL)
            return 0;
        return 1;
    }
}

```

Example 2:

```
int dict_add(dict d, char *key, char *value) {  
    // ...  
  
    // copy value to heap  
    v = strdup(value);  
    if(!v) {  
        fprintf(stderr, "could not allocate memory for value...\n");  
        return 0;  
    }  
}
```

Example 3:

```
dict dict_new() {  
    unsigned int i = 0;  
    dict d = (dict)NULL;  
    struct dict_node **c = (struct dict_node **)NULL;  
  
    // allocate memory and check for failure  
    d = (dict)malloc(sizeof(struct dict_node));  
    c = (struct dict_node **)malloc(sizeof(struct dict_node *)*MAXCHAR);  
  
    if(!d || !c) {  
        fprintf(stderr, "failed to allocate memory for dictionary...\n");  
        exit(EXIT_FAILURE);  
    }  
}
```

Example 4:

```
//this will create a new dict of 1000 entries.  
dict dict_new(){  
    dict newDict;  
    int i;  
    //allocate memory for the new dictionary  
    if((newDict = malloc(sizeof(dict))) == NULL){  
        return NULL;  
    }  
    ///allocate memory for the hash table used by the dict  
    if((newDict->table = malloc(sizeof(entry_t *) * 1000)) == NULL){  
        return NULL;  
    }  
    //initialize the table  
    for(i = 0; i < 1000; i++){  
        newDict->table[i] = NULL;  
    }  
    //set the size of the new table  
    newDict->size = 1000;  
    return newDict;  
}
```

Example 5:

```
dict_t *dict_new(int size)
{
    dict_t *new_table;

    int i = 0;

    if (size<1) return NULL; /* invalid size for table */

    /* Attempt to allocate memory for the table structure */
    if ((new_table = malloc(sizeof(dict_t))) == NULL) {
        return NULL;
    }

    /* Attempt to allocate memory for the table itself */
    if ((new_table->table = malloc(sizeof(list_t *) * size)) == NULL) {
        return NULL;
    }

    /* Initialize the elements of the table */
    for(i=0; i<size; i++) new_table->table[i] = NULL;

    /* Set the table's size */
    new_table->size = size;

    return new_table;
}
```

Example 6:

```
dict dict_new(void)
{
    dict d = (dict)malloc(sizeof(dict));
    d->hash_table = malloc(INITIAL_SIZE * sizeof(list_node));
    d->entries = 0;
    d->size = INITIAL_SIZE;
    return d;
}
```

Example 7:

```
void dict_free(dict d)
{
    free(d);
}
```

Example 8:

```
struct bucket {
    char *key;
    char *value;
    bucket_t *next;
};

struct dictionary {
```

```

    bucket_t *buckets;
};

void dict_free(dict d)
{
    int i;
    for (i = 0; i < BUCKET_COUNT; i++)
        while (d->buckets[i].next != NULL)
            free(d->buckets[i].next);
    free(d->buckets);
    free(d);
}

```

Example 9:

```

int rehash(dict d)
{
    int old_size = d->size;
    int new_size = EXPANSION_FACTOR * old_size;
    list_node **old_table = (*d).hash_table;
    list_node **new_table = malloc(new_size * sizeof(list_node));
    if (!new_table) { return -1; }
    (*d).entries = 0;
    (*d).size = new_size;
    (*d).hash_table = new_table;
    list_node *current_node;

    int i;
    for (i = 0; i < old_size; i++) {
        current_node = old_table[i];
        while (current_node) {
            dict_add(d, current_node->key, current_node->val);
            current_node = current_node->next;
        }
    }
}

```

Example 10:

```

dict dict_new(){
    dict d;
    d.hash = g_hash_table_new(g_str_hash, g_str_equal);
    return d;
}

dict_free(dict d){
    free(d.hash);
}

```

Example 11:

```

int dict_add(dict d, char *key, char *value){
    g_hash_table_insert(d.hash, key, value);

    if (value == g_hash_table_lookup(d.hash, key))
        return 1;
    else
        return 0;
}

```

Example 12:

```

static hash_size def_hashfunc(const char *key)
{
    hash_size hash=0;
    while(*key) hash+=(unsigned char)*key++;
    return hash;
}

```

Example 13:

Some General Issues

- Most of you found it way too long.
- Many of you had great difficulty.