15-122: Principles of Imperative Computation

Recitation 21b Solutions

Josh Zimmerman

0. Function pointers

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "lib/contracts.h"

typedef void* elem;
typedef int (*compare_fun)(elem x, elem y);
int linsearch_min(elem *A, int lower, int upper, compare_fun elem_compare) {
   REQUIRES(A != NULL && 0 <= lower && lower < upper && elem_compare != NULL);
   int min_idx = lower;
   for (int i = lower + 1; i < upper; i++)
      if ((*elem_compare)(A[i], A[min_idx]) < 0)
         min_idx = i;
   return min_idx;
}

int compare_strings(elem x, elem y) {
   return strcmp((char*)x, (char*)y);
}

int main(int argc, char **argv) {
   if (argc < 2) {
      fprintf(stderr, "Not enough arguments\n");
      return 1;
   }

   int min = linsearch_min((void**)argv, 1, argc, &compare_strings);

   printf("The lowest-valued command line argument was %s\n", argv[min]);
   return 0;
}
```

1. Hashtables

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "lib/xalloc.h"
#include "lib/contracts.h"
#include "lib/ht.h"

// Question 1 data structures and functions

struct q1 {
   char *word; // We won't treat struct as 'owning' this string
   unsigned int count;
};

ht_key elem_key1(ht_elem e) {
   struct q1 *S = (struct q1*)e;
   return (ht_key)S->word;
}
```

```
19
20 bool key_equal1(ht_key k1, ht_key k2) {
21    return strcmp((char*)k1, (char*)k2);
22 }
23
24 size_t key_hash1(ht_key k) {
25    // Java-style string hashing
26    size_t hash = 0;
27    char *s = (char*)k;
28    while (!*s) {
29       hash = hash*31 + *s;
30       s++;
31    }
32    return hash;
33 }
34
35 // Question 2 data structures and functions
36
37 struct q2 {
38    unsigned int key;
39    unsigned int value;
40 };
41
42 ht_key elem_key2(ht_elem e) {
43    struct q2 *S = (struct q2*)e;
44    unsigned int *p = &S->key;
45    return (ht_key)p;
46 }
47
48 bool key_equal2(ht_key k1, ht_key k2) {
49    unsigned int i1 = *(unsigned int*)k1;
50    unsigned int i2 = *(unsigned int*)k2;
51    return i1 == i2;
52 }
53
54 size_t key_hash2(ht_key k) {
55    // LCG, may not work well for 64-bit size_t
56    size_t hash = *(unsigned int*)k;
57    return hash*1664525 + 1013904223;
58 }
59
60 int main() {
61    ht H1 = ht_new(10000, &elem_key1, &key_equal1, &key_hash1, &free);
62    assert(ht_lookup(H1, "nothere") == NULL);
63    ht_free(H1);
64
65    ht H2 = ht_new(10000, &elem_key2, &key_equal2, &key_hash2, &free);
66    unsigned int x = 100;
67    assert(ht_lookup(H2, &x) == NULL);
68    ht_free(H2);
69
70    printf("All tests passed!\n");
71    return 0;
72 }
```