

0. Function pointers

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <string.h>
4 #include "lib/contracts.h"
5
6 typedef void* elem;
7 typedef int (*compare_fun)(elem x, elem y);
8 int linsearch_min(elem *A, int lower, int upper, compare_fun elem_compare) {
9     REQUIRES(A != NULL && 0 <= lower && lower < upper && elem_compare != NULL);
10    int min_idx = lower;
11    for (int i = lower + 1; i < upper; i++)
12        if ((*elem_compare)(A[i], A[min_idx]) < 0)
13            min_idx = i;
14    return min_idx;
15 }
16
17 int compare_strings(elem x, elem y) {
18     return strcmp((char*)x, (char*)y);
19 }
20
21 int main(int argc, char **argv) {
22     if (argc < 2) {
23         fprintf(stderr, "Not enough arguments\n");
24         return 1;
25     }
26
27     int min = linsearch_min((void**)argv, 1, argc, &compare_strings);
28
29     printf("The lowest-valued command line argument was %s\n", argv[min]);
30     return 0;
31 }
```

1. Hashtables

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <string.h>
4 #include "lib/xalloc.h"
5 #include "lib/contracts.h"
6 #include "lib/ht.h"
7
8 // Question 1 data structures and functions
9
10 struct q1 {
11     char *word; // We won't treat struct as 'owning' this string
12     unsigned int count;
13 };
14
15 ht_key elem_key1(ht_elem e) {
16     struct q1 *S = (struct q1*)e;
17     return (ht_key)S->word;
18 }
```

```

19
20 bool key_equal1(ht_key k1, ht_key k2) {
21     return strcmp((char*)k1, (char*)k2) == 0;
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 }

```