15-122: Principles of Imperative Computation, Fall 2014
Lab 7: Hashing Strings

Tom Cortina (tcortina@cs) and Rob Simmons (rjsimmon@cs)

Monday, October 13, 2014

For this lab, you will show your TA your answers once you complete your activities. You are expected to complete at least the first three exercises for full lab credit; exercise 4 is highly recommended.

Partial ASCII Table

32	␣	64	@	96	‘
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	’	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

1 Hashing with Strings: Finding Collisions

Recall that a hash function $h(k)$ takes a key k as its argument and returns an index for use in a hash table. In this lab you will be using various hash functions on strings and examining possibilities for collisions.

Let string s of length n ($n > 0$) be denoted as $s_0 s_1 s_2 ... s_{n-2} s_{n-1}$, where s_i is the ASCII value of character i in string s. (A partial ASCII table is given to the right.) We define five hash functions as follows:

`hash_len`: $h(s) = n$

`hash_add`: $h(s) = s_0 + s_1 + s_2 + ... + s_{n-2} + s_{n-1}$

`hash_mul32`:
$h(s) = (...((s_0 * 32 + s_1) * 32 + s_2) * 32 \ ... + s_{n-2}) * 32 + s_{n-1}$

`hash_mul31`:
$h(s) = (...((s_0 * 31 + s_1) * 31 + s_2) * 31 \ ... + s_{n-2}) * 31 + s_{n-1}$

`hash_lcg`:
$h(s) = f(f(...f(f(f(s_0) + s_1) + s_2) \ ... + s_{n-2}) + s_{n-1})$
where $f(x) = 1664525 * x + 1013904223$

Each hash function has been implemented for you and can be run from the command line. For example:

```
$ hash_len foo bar snafu
Hashing 3 strings
"foo"
   hash value = 3
   hashes to index 3 in a table of size 1024
"bar"
   hash value = 3
   hashes to index 3 in a table of size 1024
"snafu"
   hash value = 5
   hashes to index 5 in a table of size 1024
```

Note that the command line function also reports where the element with the given key will hash given a table size of 1024.

Exercise 1. Find three or more strings, each string containing three or more characters, that would always collide because they have the same hash value using `hash_add`.

Exercise 2. Find three or more strings, each string containing three or more characters, that would always collide because they have the same hash value using `hash_mul32`. (Hint: use the fact that 32 is a power of 2.)

Exercise 3. Find three or more strings, each string containing three or more characters, that would always collide because they have the same hash value using `hash_mul31`.

2 Programming

Exercise 4. Implement your own version of `hash_mul31` as a function that takes a single non-empty string as its argument and returns an integer representing the hash value for that string using the formula given on the previous page. Demonstrate that it works correctly by comparing the results of this function in `coin` with your answers from Exercise 3. Your function does not need to compute the hash index for a table of size 1024. (Fun fact: Java uses this algorithm for computing hash values.)

Exercise 5. CHALLENGE Using `hash_mul31`, find three or more strings, each containing three or more characters, that do not have the same exact hash values but do collide in a hash table of size 1024.

Exercise 6. CHALLENGE Find three or more strings, each containing three or more characters, that collide in a hash table of size 1024 using `hash_lcg`. HINT: Write some code to help you find the strings. Why aren't we asking you to find hash values that match exactly?