CJam 0.6.5 cheat sheet

Integer Arithmetic

`+`	add	`-`	subtract[1]
`*`	multiply	`/`	divide
`%`	modulo	`#`	power
`(`	−1	`)`	+1
`m!`	factorial	`md`	divmod
`mf`	factors (`mF` → exponents)		
`mp`	is prime?	`mQ`	integer √
`z`	absolute value		
`ex`	multiply by 10^x (literal *x*)		

Bitwise Arithmetic

`~`	negate	`|`	bit-or
`&`	bit-and	`^`	bit-xor
`m<`	left shift	`m>`	right shift

Logic

`<`	less than	`>`	greater
`e<`	min[2]	`e>`	max
`=`	equals	`!`	logical not
`e&`	and	`e|`	or

Float Arithmetic

`ms`	sine	`mS`	arcsin
`mc`	cosine	`mC`	arccos
`mt`	tangent	`mT`	arctan
`me`	e^x	`ma`	atan2
`ml`	ln	`mL`	$\log_b$
`m[`	floor	`m]`	ceil
`mo`	round	`mO`	round to
`mq`	$\sqrt{x}$	`mh`	$\sqrt{x^2+y^2}$

Stack operations

`;`	discard	`a→`	
`\`	swap	`ab→ba`	
`_`	duplicate	`a→aa`	
`@`	rotate	`abc→bca`	
`$`	copy *n*th	`…s1s0n→sn`	

Control Flow

`%`	map	`/`	each
`&`	do-if	`|`	do-unless
`g`	do-while	`h`	do-while'[3]
`?`	ternary if	`~`	eval
`*`	repeat block; reduce		
`,`	filter	`w`	while
`fV`	for loop in variable *V*		

Arrays[4]

`[]`	begin/end	`~`	dump
`a`	wrap in array		
`=`	get, find	`t`	set
`<`	take	`>`	drop
`+`	concat	`*`	repeat/join
`/`	split	`%`	clean split
`,`	range, len	`$`	sort (by)
`&`	intersect	`|`	union
`^`	symmetric difference		
`-`	remove	`e_`	flatten
`#`	find index	`z`	transpose
`(`	uncons lt	`)`	uncons rt
`m<`	rotate lt	`m>`	rotate rt
`e!`	permutations (no reps)		
`m!`	permutations (reps)		
`e*`	repeat all	`e=`	count
`e[`	pad left	`e]`	pad right
`e\`	swap two indices		
`e``	run-length encode		
`e~`	run-length decode		
`ee`	enumerate: $[a_i] \to [(i, a_i)]$		
`ew`	overlapping slices		
`m*`	cartesian product/power		

Conversions

`i`	integer	`d`	double
`s`	string	`c`	char
`` ` ``	string representation		
`b`	base conversion		

Strings

`e%`	format	`er`	translate
`el`	lowercase	`eu`	uppercase

Input/Output

`r`	read token (space-sep.)		
`l`	read line	`o`	print
`q`	read all	`p`	print line
`ea`	command line args		

Miscellaneous

`mr`	random number, shuffle		
`mR`	random choice		
`et`	time[5]	`es`	unix time
`ea`	command line args		
`ed`	debug (stack contents)		
`j`	memoize/recurse		
`fg`	map *g* with param.		

1 Immediately before integer literals, you can use `m` instead of `-`, to avoid having to insert whitespace to parse e.g. `AB-3+` correctly.

2 The operators `e<` and `e>` pop *two arguments* to compare, not a list.

3 `h` leaves the condition on the stack at the end of a loop; `g` pops it.

4 Array functions that don't have some other meaning when applied to ints will try to convert an integer argument *n* to an array *[0, 1... n-1]*.

5 [y m d h m s ms weekday timezone]

Shortcuts: `.g` – *zipWith* (*g* a binary op or block). `:V` – assignment. `:g` – short map/fold. **Vars:**

A	B	…	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
10	11	…	20	[]	[]	\n	[]	π	[]	[]	spc	0	0	0	-1	1	2	3