Summary Page: Pointers, Arrays & Strings
CISC 220, Fall 2012

To create an array of 10 integers:

`int nums[10];`

To declare a pointer to an integer:

`int *ptr;`

Operators related to pointers:

`&x` = the address of `x`

`*ptr` dereferences `ptr` (finds the value stored at address `ptr`)

Using the heap:

`malloc(n)`: returns a pointer to n bytes on the heap

`free(ptr)`: releases heap space, where `ptr` is the result of a call to `malloc`

Type sizes:

`sizeof(typ)`: returns the number of bytes used by a value of type `typ`

Creating strings:

`char abbrev[] = "CISC";` // array containing 5 characters (for CISC plus `'\0'`)

`char abbrev[10] = "CISC";` // 10 characters, the first 5 initialized to CISC plus `'\0'`

`printf` conversions for printing strings:

`%s`: print the whole string, no padding

`%20s`: print the whole string with a minimum length of 20, padding on the left if necessary

`%-20s`: print the whole string with a minimum length of 20, padding on the right if necessary

`%.5s`: print the whole string with a maximum length of 5, truncating if necessary

Printing strings with `puts`:

`puts(str)`: writes `str` to the standard output, followed by `'\n'`

Reading strings:

`scanf("%20s", str):` skips white space, then reads characters until either it reaches 20 characters or it gets to more white space or the end of the file. The 20 characters doesn't count the ending `'\0'` – so `str` must have room for at least 21.

`fgets(str, 21, stdin)`: reads into `str` until it has 20 characters or it reaches the end of the line. `str` will have `'\n'` at the end unless there were 20 or more characters in the line.

More useful string functions:

`strlen(str)`: returns the length of `str` (not counting the ending `'\0'`)

`strcpy(s1,s2)`: copies contents of `s2` to `s1`

`strncpy(s1,s2,n)`: copies at most `n` characters from `s2` to `s1`
(no guarantee of ending '`\0'`)

`strcat(s1,s2)`: concatenates `s2` to end of `s1`

`strcmp(s1,s2)`: returns an integer:

`0` if `s1` and `s2` are equal

negative if `s1` < `s2` (i.e. `s1` would come first in a dictionary)

positive if `s1` > `s2`

Converting from string to integer:

`long int strtol(char *string, char **tailptr, int base)`

returns `string` converted to an integer

`base` should be the radix, normally 10

`tailptr` should be the address of a pointer

`strtol` will set `*tailptr` to the address of the first character in `string` that wasn't used