

# CS 3: Introduction to Software Design

## Pointers Exercises

### Warmup

fread Prototype Reminder

```
size_t fread(void *ptr, size_t size, size_t nitems, FILE *stream);
```

### Fill In The Blanks!

```
1 void read_one_1() {  
2     Char * c = malloc(sizeof(char));  
3     fread(c, sizeof(char), 1, stdin);  
4     printf("I got: %c\n", *c);  
5 }  
  
1 void read_one_2() {  
2     Char c = 'X';  
3     fread(&c, sizeof(char), 1, stdin);  
4     printf("I got: %c\n", c);  
5 }
```

Handwritten annotations for read\_one\_1:

- Line 2: A box labeled "Char \*". A green arrow points from this box to the variable "c" in line 2.
- Line 3: A box labeled "C". A green arrow points from this box to the variable "c" in line 3.
- Line 4: A box labeled "\*c". A green arrow points from this box to the argument of the printf function.
- Line 5: A box labeled "H'd \rightarrow". A green arrow points from this box to the value of "c" in line 4.

Handwritten annotations for read\_one\_2:

- Line 2: A box labeled "Char". A green arrow points from this box to the variable "c" in line 2.
- Line 3: A box labeled "d". A green arrow points from this box to the variable "c" in line 3.
- Line 3: A box labeled "(char \*)". A green arrow points from this box to the variable "c" in line 3.
- Line 3: A box labeled "Char". A green arrow points from this box to the variable "c" in line 4.
- Line 4: A box labeled "Char \* c ;". A green arrow points from this box to the argument of the printf function.
- Line 5: A box labeled "((char \*) c) \rightarrow ((char \*) c)". A green arrow points from this box to the value of "c" in line 4.
- Line 5: A box labeled "((char \*) c) \rightarrow ((char \*) c)". A green arrow points from this box to the value of "c" in line 4.
- Line 5: A box labeled "((char \*) c) \rightarrow ((char \*) c)". A green arrow points from this box to the value of "c" in line 4.

### More &

```
1 int to_nibble(char *bin) {  
2     char *endptr = NULL;  
3     char *dup = strndup(bin, 4);  
4     int result = strtol(dup,  , 2);  
5     if (  != endptr) {  
6         free(dup);  
7         return -1;  
8     }  
9     free(dup);  
10    return result;  
11 }
```

```
long strtol(char *str, char **endptr, int base)  
-----  
If endptr is not NULL, strtol() stores the  
address of the first invalid character in  
*endptr. If there were no digits at all,  
however, strtol() stores the original value of  
str in *endptr. (Thus, if *str is not '\0' but  
**endptr is '\0' on return, the entire string  
was valid.)
```

## Pointer Arithmetic

```
1 int strip_leading_zeroes(char **ptr) {  
2     int result = 0;  
3     while ((*ptr)[0] == '0' && (*ptr)[1] != '\0') {  
4         /* Boxed code */  
5         /* Boxed code */  
6     }  
7     return result;  
8 }
```

## Bonus

```
1 char *pad_to_n(char *str, size_t n) {  
2     size_t len = (strlen(str) / n + 1) * n;  
3     char *out = calloc(len + 1, sizeof(char));  
4     char *ptr = out;  
5     for (size_t i = 0; i < len - strlen(str); i++) {  
6         *ptr = '0';  
7         ptr++;  
8     }  
9     while (/* Boxed code */){  
10        /* Boxed code */  
11        ptr++;  
12        str++;  
13    }  
14    return out;  
15 }
```

## All Together Now...

```
1 char *readline(char **buf) {  
2     int charsread = 0;  
3     char c = '\n';  
4     while (fread(/* Boxed code */, sizeof(char), 1, stdin) && c != '\n') {  
5         charsread++;  
6         /* Boxed code */  
7     }  
8 }  
9 return /* Boxed code */;  
10 }
```