C - POINTERS, FUNCS, & INPUT

FUNCTIONS AND POINTERS

- If you want to be able to change the variable:
  - pass a pointer
  - function declaration must specify argument is pointer
- Passing arrays
  - Always passes a pointer
  - Typically pass size as separate argument
- Returning arrays
  - Either return pointer (or modify in place)

MEMORY ALLOCATION (CONT.)

- Can also allocate memory for just a single value
- Example:

```
int *iptr;
iptr = (int *) malloc(sizeof(int));
```

READING FROM STDIN

- `scanf(formatstr, memaddr1, memaddr2, ...)`
  - `formatstr` is the same type of format string used for `printf()`
  - Every `%d`, `%f`, `%s`, etc. used in `formatstr` needs a memory address
  - If EOF signal (Ctrl-D) is sent, scanf returns 0 or -1 (depends on implementation)
    - value is in EOF macro in stdio.h
    - check to see if return result equals EOF

READING FROM STDIN

- `fgets(char *s, int size, FILE *stream)`
- for stdin, `FILE *stream` should be `stdin`
  - we'll get to other File I/O later
- need to allocate memory yourself
- will only read and store at most n-1 characters
- null character added after last character read (`\0`)
- will not read beyond newline or EOF
- returns `s` if successful, `NULL` if unsuccessful

READING FROM STDIN

- `getline(char **lineptr, size_t *n, FILE *stream)`
- technically not in C standard - part of POSIX > 2008
- will reallocate memory if there is not enough room to store whole line
- if `*lineptr` is NULL and `n=0`, will allocate memory
- `*lineptr` and `*n` will be updated after
- returns number of characters read, or -1 if it errors

MINILAB 11