# CS 300
# Stacks via List

# gcc

- The `-I` option tells gcc where to look for extra header files:

```
gcc -I include src/test.c -o bin/test.o
gcc -o bin/test bin/test.o bin/stack.o

test.c:
#include <stdio.h>
#include “stack.h” // no need for ../include

int main()
{
```

# extern vs static

- **`extern`** in C means externally defined
  - With a function: give this function visibility outside the current module (.c file)
  - All functions have extern added implicitly
  - With a variable: don't allocate memory for this variable: the variable is defined somewhere else

- **`static`** in C has two meanings
  - Make a variable/function non-global
  - Make a local variable in a function retain its value

# extern

```
// global variable declared
// in some other .c file

test.h:
extern int gValue;

test.c:
#include "test.h"
int gValue;

// what happens if we
// declare gValue in
// the .h file?

driver.c
#include "test.h"
// has access to gValue during
// gcc -o exe bin/driver.o bin/test.o
```

# static

```
test.c:
static int gValue;

driver.c
#include “test.h”
// has NO access to gValue during
// gcc -o exe bin/driver.o bin/test.o

int counter()
{
  static int count = 0;
  count ++;
  return count;
}
```

# Stack

- stkCreate()
- stkPush()
- stkPop()
- stkPeek()

```
Typedef struct
Stack
{
   List sTheList;
   // ??
   // ??
} Stack;
```