Midterm Review

- Practice material from the Winter 2014 midterm.
- This material only covers up to modules.
- Midterm content: Everything up to / including arrays.
Short Answer

List three advantages of modularization, and briefly explain.
Short Answer

List three advantages of modularization, and briefly explain:

- **Re-usability**: Build programs faster, and build large programs more easily.

- **Maintainability**: Makes debugging and changing a program easier.

- **Abstraction**: No need to know / understand all parts to use them. Allows changing internals without breaking things.
Clicker Question

Every C module must have a main function.

A  True.
B  False.
C  True, but only on Mondays.
Clicker Question

Every C module must have a main function.

A True.
B *False.
C True, but only on Mondays.

A module does not require a main function. (Example: The stack module from the last assignment).
Clicker Question

In C, \((a \neq 0) \&\& (b/a == 2)\) will produce an error if \(a\) is 0.

A  True.
B  False.
C  Impossible to know without trying it.
Clicker Question

In C, \((a \neq 0) \&\& (b/a == 2)\) will produce an error if \(a\) is 0.

A  True.
B  *False.
C  Impossible to know without trying it.

Because \(\&\&\) is used, the first condition will short circuit the evaluation, preventing \(b/a\) from being executed.
Short Answer

Briefly explain the purpose of *requires* and *effects* function documentation in C.
Briefly explain the purpose of \textit{requires} and \textit{effects} function documentation in C:

- \textbf{Requires}: Identifies conditions that must be true when calling the function (e.g. restrictions on parameters).
- \textbf{Effect}: Identifies if a function prints, reads, mutates a global variable or mutates a variable through a pointer as a parameter.
Clicker Question

printf("hello!\n") is a C expression with a value of 7.

A  True.
B  False.
C  The only “expressions” in C are my cries of pain when I’m forced to code in it.
Clicker Question

`printf("hello!\n")` is a C expression with a value of 7.

A  *True.
B  False.
C  The only “expressions” in C are my cries of pain when I’m forced to code in it.

The return value of `printf` is the number of characters printed.
Short Answer

Write the declaration for a C function add that takes two ints, x and y, and returns an int x+y.
Short Answer

Write the declaration for a C function add that takes two ints, x and y, and returns an int x+y:

- No actual addition required!
- `int add(int x, int y);`
Clicker Question

In the following C code, the assignment operator appears only once:

```c
bool nisfive = (n == 5);
```

A  True.
B  False.
C  This has to be harder than it looks but I’m not seeing the trick.
Clicker Question

In the following C code, the assignment operator appears only once:
bool nisfive = (n == 5);

A  True.
B  *False.
C  *This has to be harder than it looks but I’m not seeing the trick.

The code above uses = for initialization, NOT assignment. They are very slightly different (e.g. some struct-related notation can only be used during initialization, and not assignment).
Coding Question

Write the C function pyramid(int n); that prints a pyramid of numbers with n lines. You may assume n is an odd integer in the range [3-99]. Use the printf formatter “%3d” to print out each integer. pyramid(7) prints the following:

1
2 3
3 4 5
4 5 6 7
3 4 5
2 3
1
Coding Question

General approach:

- Need nested for loops / while loops.
- One set of loops to count from row 1 to $n/2 + 1$, one set to count back down from row $n/2$ to 1.
void pyramid(int n) {
    for (int i=1; i<=n/2+1; ++i) {
        for (int j=i; j<=2*i-1; ++j) {
            printf("%3d",j);
        }
        printf("\n");
    }
    for (int i=n/2; i>0; --i) {
        for (int j=i; j<=2*i-1; ++j) {
            printf("%3d",j);
        }
        printf("\n");
    }
}
Stack Trace

Draw the diagram that shows the stack frame after line 3 is executed.

```c
int main(void) {
    int i = 42;
    int * p = &i;
}
```