UNIVERSITY OF MASSACHUSETTS
DARTMOUTH

ECE160: Foundations of Computer Engineering I

Lecture #11 – Exam #1 Review

Instructor: Dr. Liudong Xing
SENG-213C, lxing@umassd.edu
ECE Dept.

Exam #1

- Time: **9:00am ~ 10:30am, Friday, Feb. 17**
- Please arrive at the class on time; no make up time will be given for late arrivals.
- Form:
  - Open book open notes
  - Calculators are NOT allowed
- Preparation:
  - Lecture notes #2 - #10
  - Homework #1 - #2
  - Lab #2 - #4

Lectures #2 - #10

- Number systems (L#2)
- Introduction to C programming (L#3)
- Data types and variables (L#4)
- Constants (L#5)
- Formatted input/output (L#6 & 7)
- Expressions (L#8 & 9)
- Two-way selection: if…else (L#10)

Number Systems (L#2)

1. Basic number systems concepts (base, positional/place value, symbol value)
2. How to work with numbers represented in binary, octal, and hexadecimal number systems
3. How to convert back and forth between decimal numbers and their binary, octal, and hexadecimal equivalents
4. How to abbreviate binary numbers as octal or hexadecimal numbers
5. How to convert octal and hexadecimal numbers to binary numbers

Number Systems (L#2: 1)

- Basic number systems concepts
  - Base: determines the magnitude of a place
  - positional/place value: power of the base
  - symbol value:digit *x* positional value

Example: consider decimal number 4538

- Base: **10**
- The positional value of digit 5: **10^2**
- The symbol value of digit 5: **$5 \times 10^2 = 500$**

Number Systems (L#2: 2)

- Binary (base 2)
  - 0,1
- Octal (base 8)
  - 0,1,2,3,4,5,6,7
- Decimal (base 10)
  - 0,1,2,3,4,5,6,7,8,9
- Hexadecimal (base 16)
  - 0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Number Systems (L#2: 3)

- Convert back and forth between decimal numbers and their binary, octal, and hexadecimal equivalents
  - To convert any base to decimal we multiply the decimal equivalent of each digit by its positional / place value (a power of the base) and sum these products
  - To convert decimal numbers to any base we divide with the corresponding base until the quotient is zero and write the remainders in reverse order.

Number Systems (L#2: 4)

- How to abbreviate binary numbers as octal or hexadecimal numbers
  - To convert from binary to Hex, simply divide the binary number into *4-bit* group (from right to left) and then write those groups over the corresponding digits of the hex number
  - To convert from binary to Octal, simply divide the binary number into *3-bit* group (from right to left) and then write those groups over the corresponding digits of the Octal number

Number Systems (L#2: 5)

- How to convert octal and hexadecimal numbers to binary numbers
  - To convert from Octal to binary, simply write each octal digit its *3-digit* binary equivalent
  - To convert from Hex to binary, simply write each hex digit its *4-digit* binary equivalent

Lectures #2 - #10

- Number systems (L#2)
- **Introduction to C programming (L#3)**
- Data types and variables (L#4)
- Constants (L#5)
- Formatted input/output (L#6 & 7)
- Expressions (L#8 & 9)
- Two-way selection: if…else (L#10)

Intro. to C Programming (L#3)

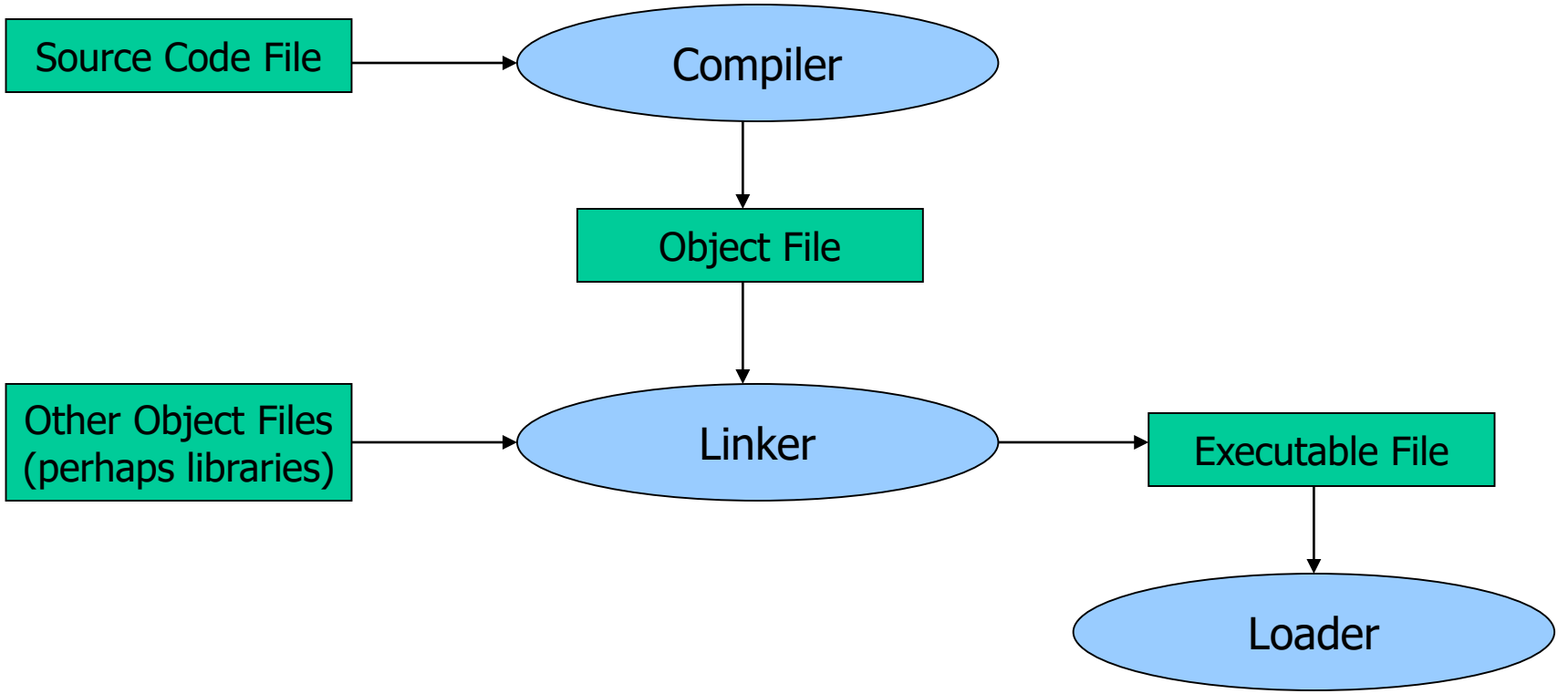
1. Computer languages evolution: machine → assembly → high-level (e.g., C) →…
2. A popular software development lifecycle – waterfall model
3. The first C program
4. Identifiers and naming rules
5. Two types of errors: syntax and logic / semantics errors

Intro. to C Programming (L#3: 1)

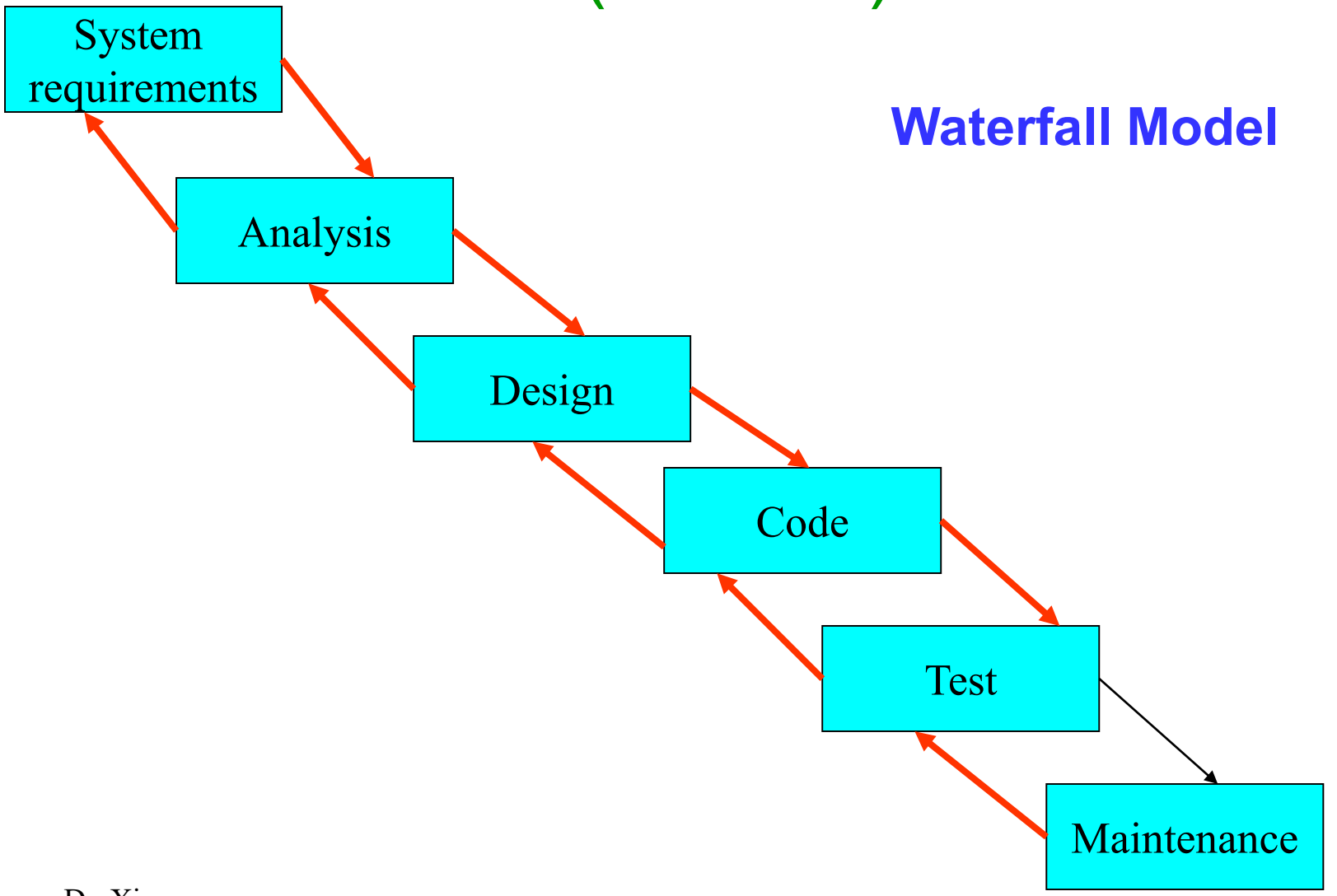
- Machine languages are binary-based code (made of streams of 0s and 1s)
  - The only language understood by computers

- High-level languages (like C) are generally machine-independent
  - Usually, several machine instructions are combined into one high-level instruction.
  - Translated into executable form using compiler and linker in C

Modern Software Development (Review)

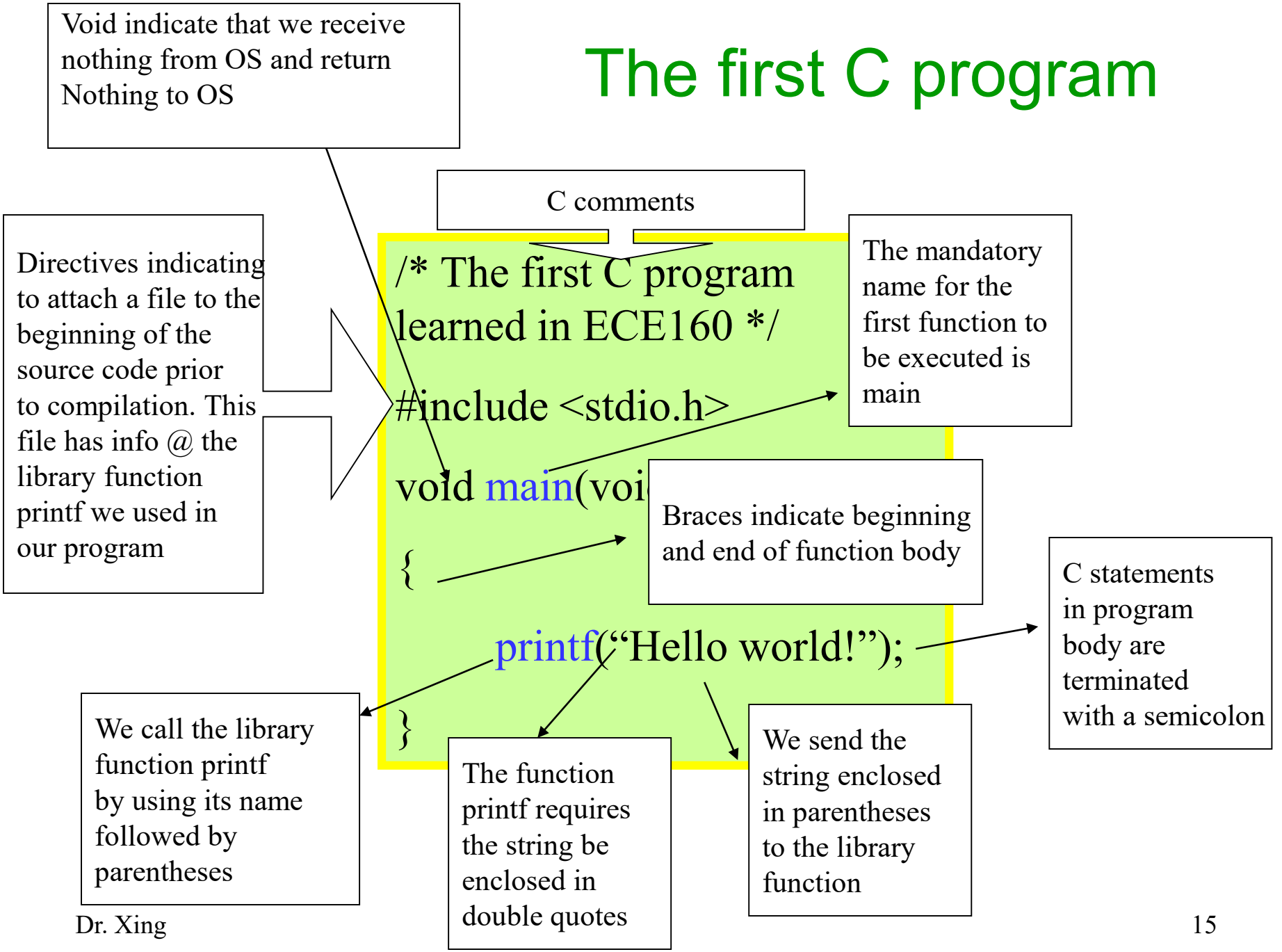


Software Development Lifecycle (Review)



Waterfall Model

The first C program



Identifiers and Naming Rules

- Identifiers are used to name data and other objects (e.g. functions) in our program.
- C is case sensitive
  - Celsius, celsius, and CELSIUS are three different identifiers.
- Rules
  - The first character can not be a digit. It has to be an alphabetic character or underscore.
  - The identifier name must consist only of alphabetic characters, digits, or underscores.
  - First 31 characters of an identifier are significant/used.
  - DO NOT use a C reserved word /keywords (e.g., **int**).

Two Types of Errors

- Syntax: the required form of the program punctuation, keywords (int, float, return, …) etc.
  - Examples:
    - putting a semicolon after main() is a compilation error
    - Forgetting to terminate a comment with */ is a compilation error.
  - The C compiler always catches these “syntax errors” or “compiler errors”
- Semantics (logic): what the program means
  - What you want it to do
  - The C compiler cannot catch these kinds of errors!
  - They can be extremely difficult to find

Lectures #2 - #10

- Number systems (L#2)
- Introduction to C programming (L#3)
- **Data types and variables (L#4)**
- Constants (L#5)
- Formatted input/output (L#6 & 7)
- Expressions (L#8 & 9)
- Two-way selection: if…else (L#10)

Standard Data Types

- void: has no values
- int: a number without fraction part
  - 3 different sizes of the integer type: short int, int, long int
  - the size of int is machine dependent
  - C supports logical data type through the integer type
- char: a value that can be represented in the computer's alphabet.
  - represented using 1 byte (ASCII code)
- float: a number with fraction part
  - 3 types of floating point numbers: float, double, long double

Variables

- Variables are named memory locations that have a type, identifier, and value.
- Each variable in the program must be declared and defined!
  - Declaration: to name a variable
  - Definition: to create a variable, to reserve memory for it
  - Usually a variable is declared and defined at the same time!
- The programmer must initialize any variable requiring prescribed data when the function starts

Lectures #2 - #10

- Number systems (L#2)
- Introduction to C programming (L#3)
- Data types and variables (L#4)
- **Constants (L#5)**
- Formatted input/output (L#6 & 7)
- Expressions (L#8 & 9)
- Two-way selection: if…else (L#10)

Constants (L#5)

- Four types: Integer (13), Character (‘a’), Floating point (2.3), String (“hello”)
- Three ways to code constants in the program:
  - Literal: an unnamed constant, the data itself (3.14)
  - Defined: use the preprocessor command define (*e.g.:* #define PI 3.14) --- the expression that follows the name replaces the name wherever it is found in the source program
  - Memory: Use a C type qualifier: *const (e.g.:* const float pi = 3.14;) --- memory constants fix the contents of a memory location

Lectures #2 - #10

- Number systems (L#2)
- Introduction to C programming (L#3)
- Data types and variables (L#4)
- Constants (L#5)
- **Formatted input/output (L#6 & 7)**
- Expressions (L#8 & 9)
- Two-way selection: if…else (L#10)

Formatted Output *printf()* (L#6: 1)

- printf(format string, data list);
  - Instructions for formatting the data, and
  - The actual data to be printed.

- Conversion codes %d %c %f etc
  - The number of conversion code should match the number of data/variables that follow the "format string"

Formatted Output *printf()* (L#6: 2)

- Field width specification: specifying the number of digits to display
  - When there are more places in the field width than digits to be displayed, the output is right-justified.
  - When there are more digits than places, the output field width is ignored, and the entire integer is displayed.
- Flag modifiers: 0 and –
  - 0: the number will be printed with leading zeros
  - - (minus sign): the data are formatted left justified
- Output special characters using \

Formatted Input *scanf()* (L#7)

- Function format
  - scanf(format string, address list);
  - The number of conversion code should match the number of addresses that follow the "format string"
  - Each variable name in the address list must be preceded by an ampersand **&**.

- You can use field width like %2d, but there is no precision width in the input field specification. When scanf() finds a precision, it stops processing.

Lectures #2 - #10

- Number systems (L#2)
- Introduction to C programming (L#3)
- Data types and variables (L#4)
- Constants (L#5)
- Formatted input/output (L#6 & 7)
- **Expressions (L#8 & 9)**
- Two-way selection: if…else (L#10)

C Expressions (1)

- Types of expressions
  - Primary expressions: consist of only one operand with no operator
  - Binary expressions: formed by an *operand-operator-operand* combination
    - Multiplicative expressions: *, /, %
    - Additive expressions: +, -
  - Assignment expressions using assignment operator =
  - Postfix expressions: a++; a--;
  - Unary expressions:
    - Prefix increment/decrement: ++a; --a;
    - Sizeof()
    - plus/minus

C Expressions (2)

- A side effect is an action that results from the evaluation of an expression: *changing the value of a variable is a side effect*
  - side effects take place before the expression is evaluated: ++a; --a;
  - side effects take place after the expression is evaluated: a++; a--;
- Precedence and associativity
  - Precedence determines the order in which different operations are evaluated.
  - Associativity determines how operators with the same precedence are grouped together in complex expressions (left, right)
  - Note that precedence is applied before associativity.

Operator Precedence (in descending order)

Postfix operators: ++, --, ..
Prefix operators: ++, --, ..
sizeof
Plus/minus signs: +,-
Logical NOT: !
Type cast: ()
Multiplicative operators: *, /, %
Addition: +, -
Shift: << , >>
Relation: < , <=, >, >= ..
Equality operations: ==, !=
Bitwise/Boolean AND: &
Bitwise/Boolean XOR: ^
Bitwise/Boolean OR: |
Logical AND: &&
Logical OR: ||
Ternary conditional operator: ?:
Assignment: = , +=, -=, etc..

C Expressions (3)

- Evaluating complex expressions
  - Expressions without side effects
  - Expressions with side effects

- Mixed type expressions
  - Implicit type conversion by compiler
    - In an assignment expression, the final expression value must have the same type as the left operand, the operand that receives the value!
    - Variables with low precedence are promoted to match the highest precedence hierarchy in the expression.
  - Explicit type conversion using type cast operator (new type) by programmers

Lectures #2 - #10

- Number systems (L#2)
- Introduction to C programming (L#3)
- Data types and variables (L#4)
- Constants (L#5)
- Formatted input/output (L#6 & 7)
- Expressions (L#8 & 9)
- **Two-way selection: if…else (L#10)**

Two-Way Selection *if…else* (L#10)

- Logical data: true (1) or false (0)
  - C supports this through int type: zero (false), non-zero (true)

- 3 logical operators:
  - ! NOT, && (logical AND), || (logical OR)

- 6 relational operators

<	less than
>	greater than
<=	less than or equal
>=	greater than or equal
==	equal
!=	not equal

Two-Way Selection *if…else* (2)

- *if…else* statement

```
if (expression)
    {
     Action 1
    }
   else
    {
   Action 2
   }
```

- Nested *if…else* statement: An *if…else* is included within another *if…else*
- Dangling *else* problem: when there is no matching *else* for every *if,* Solution: *Always pair an "else" to the most recent unpaired "if" in the current block!*
- Ternary conditional operator

  expression1 ? expression2 : expression3

  – This means that if expression1 is true, then the overall expression evaluates to expression 2, else it evaluates to expression3.

An Example

```
#include "stdafx.h"
void main(void)
{
    int a,b;
    printf("Enter two integers:\n");
    scanf("%d%d",&a, &b);
    if(a >= b)
        {
            if(a > b)
                printf("%d > %d",a,b);
            else
                printf("%d == %d",a,b);

        }
    else
        {
            printf("%d < %d", a, b);
        }
}
```

Good programming style:

Using indention

Line up opening and closing braces

Exam #1

- Time: **9:00am ~ 10:30am, Friday, Feb. 17**
- Please arrive at the class on time; no make up time will be given for late arrivals.
- Form:
  - Open book open notes
  - Calculators and computers are NOT allowed
- Preparation:
  - Lecture notes #2 - #10
  - Homework #1 - #2
  - Lab #2 - #4

Good Luck!