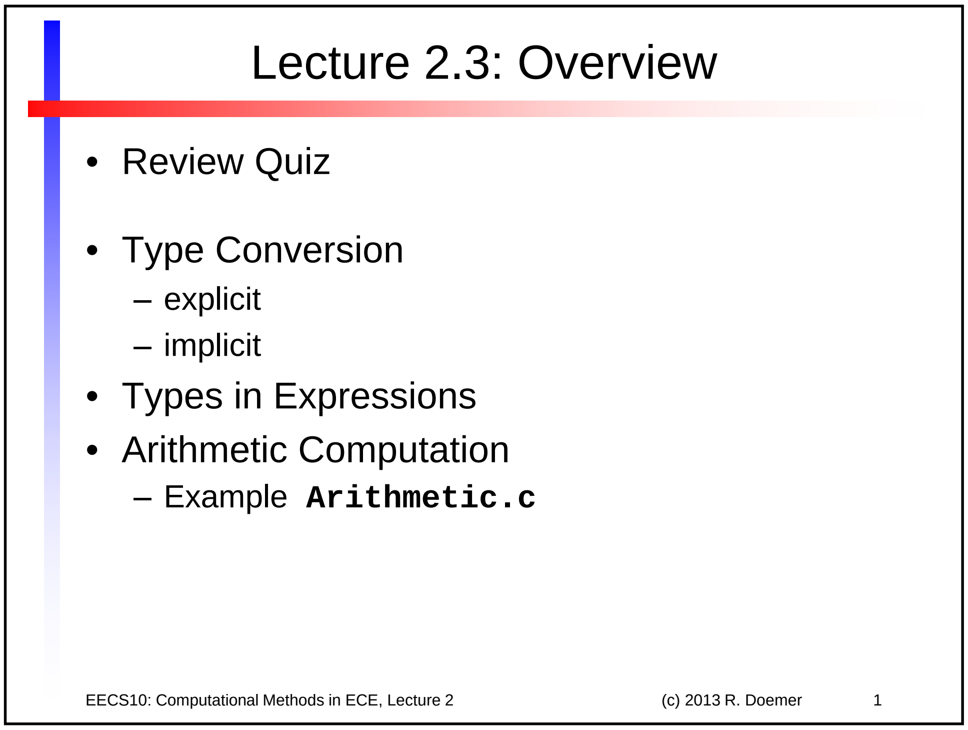
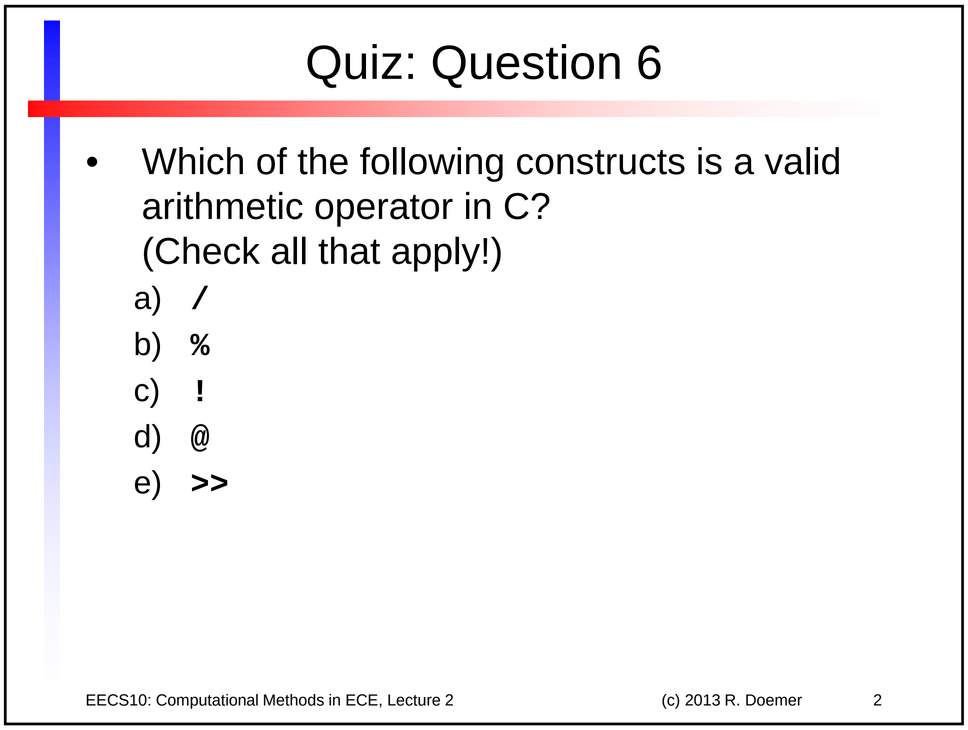


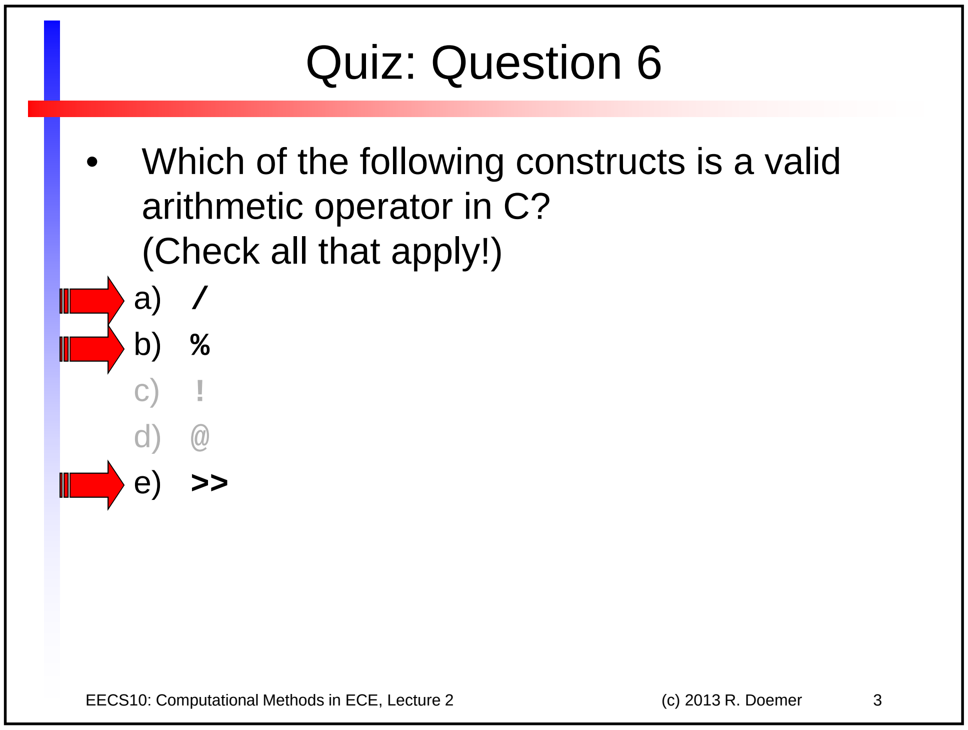
Lecture 2.3: Overview

- Review Quiz
- Type Conversion
  - explicit
  - implicit
- Types in Expressions
- Arithmetic Computation
  - Example **`Arithmetic.c`**



Quiz: Question 6

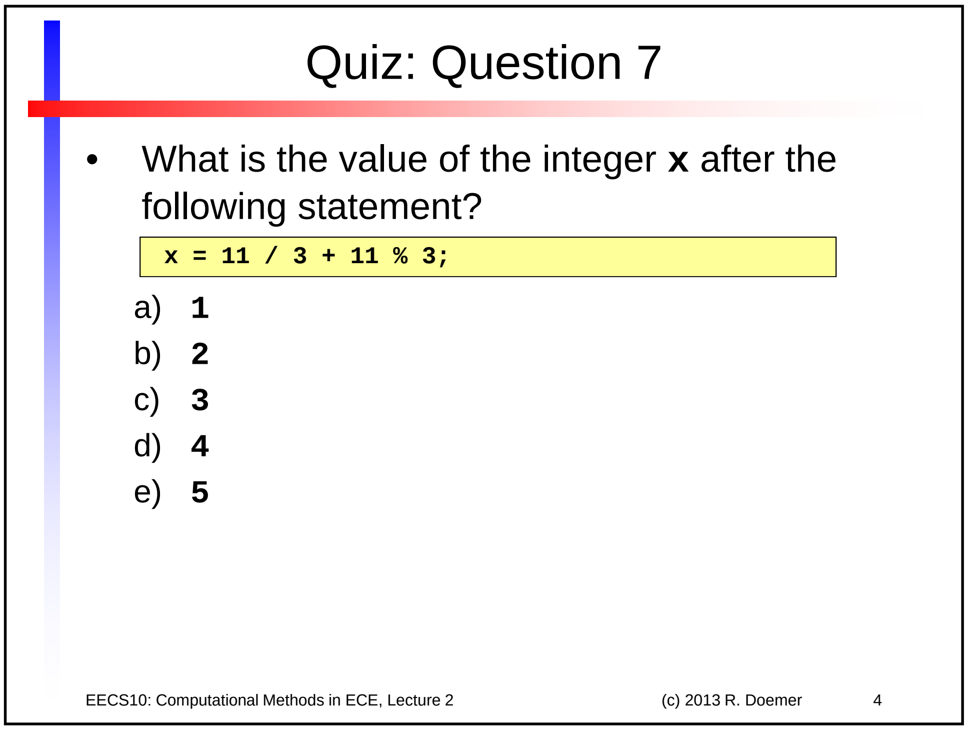
- Which of the following constructs is a valid arithmetic operator in C?
  (Check all that apply!)
  a) `/`
  b) `%`
  c) `!`
  d) `@`
  e) `>>`



Quiz: Question 6

- Which of the following constructs is a valid arithmetic operator in C?
  (Check all that apply!)

[✓ a) `/`]
[✓ b) `%`]
c) `!`
d) `@`
[✓ e) `>>`]

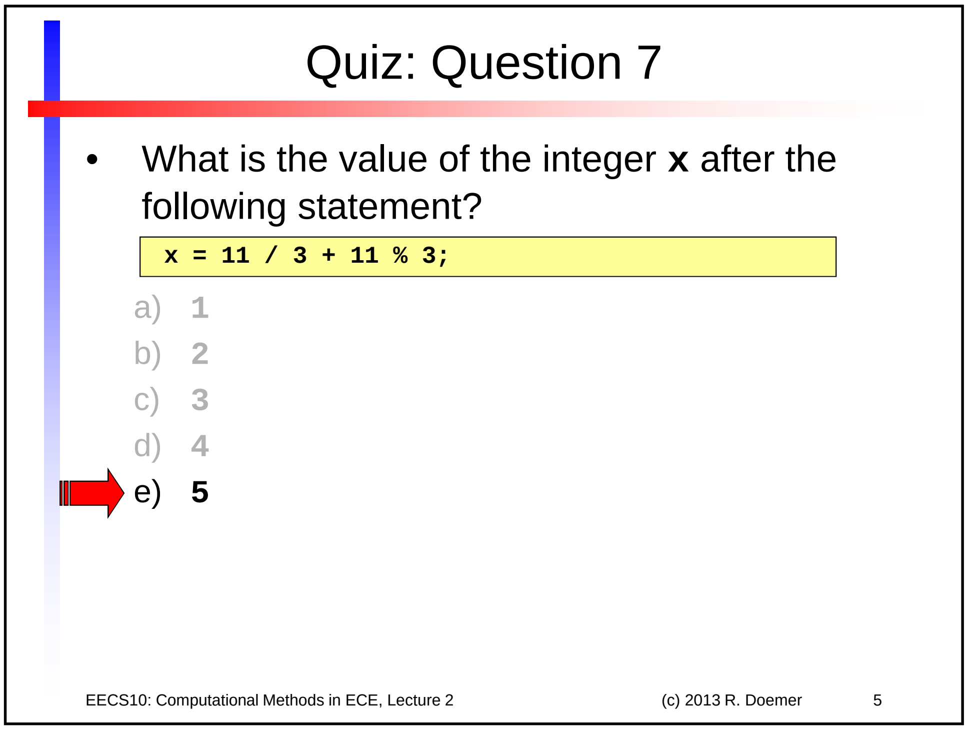


Quiz: Question 7

- What is the value of the integer **x** after the following statement?

```
x = 11 / 3 + 11 % 3;
```

a) **1**
b) **2**
c) **3**
d) **4**
e) **5**

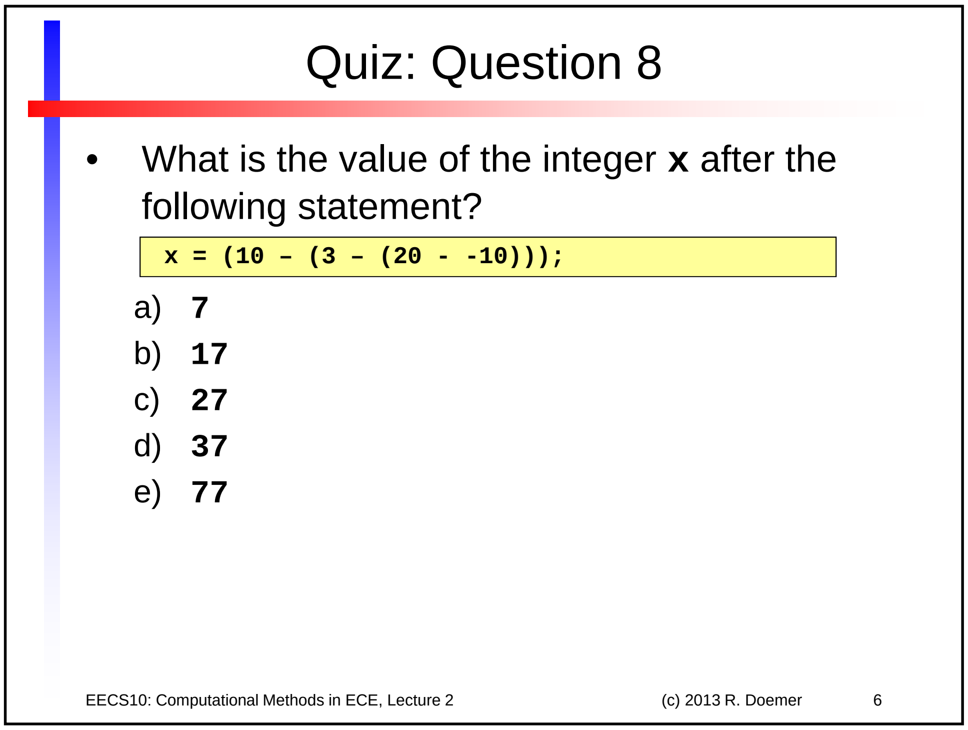


Quiz: Question 7

- What is the value of the integer **x** after the following statement?

```
x = 11 / 3 + 11 % 3;
```

a) **1**
b) **2**
c) **3**
d) **4**
[✓ e)] **5**

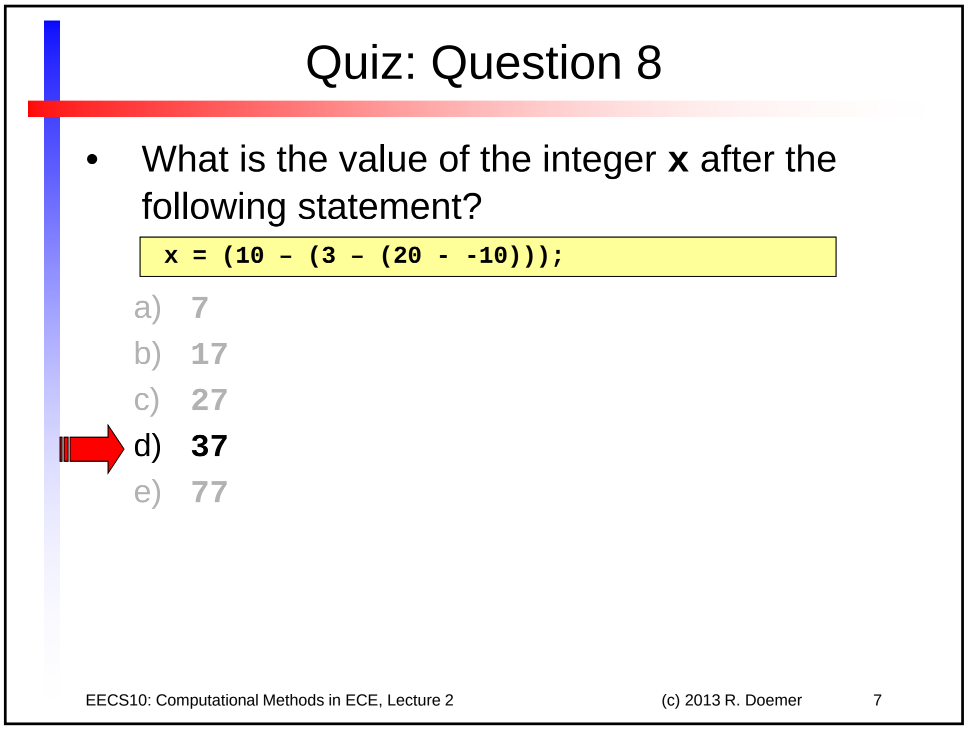


Quiz: Question 8

- What is the value of the integer **x** after the following statement?

```
x = (10 – (3 – (20 - -10)));
```

a) **7**
b) **17**
c) **27**
d) **37**
e) **77**

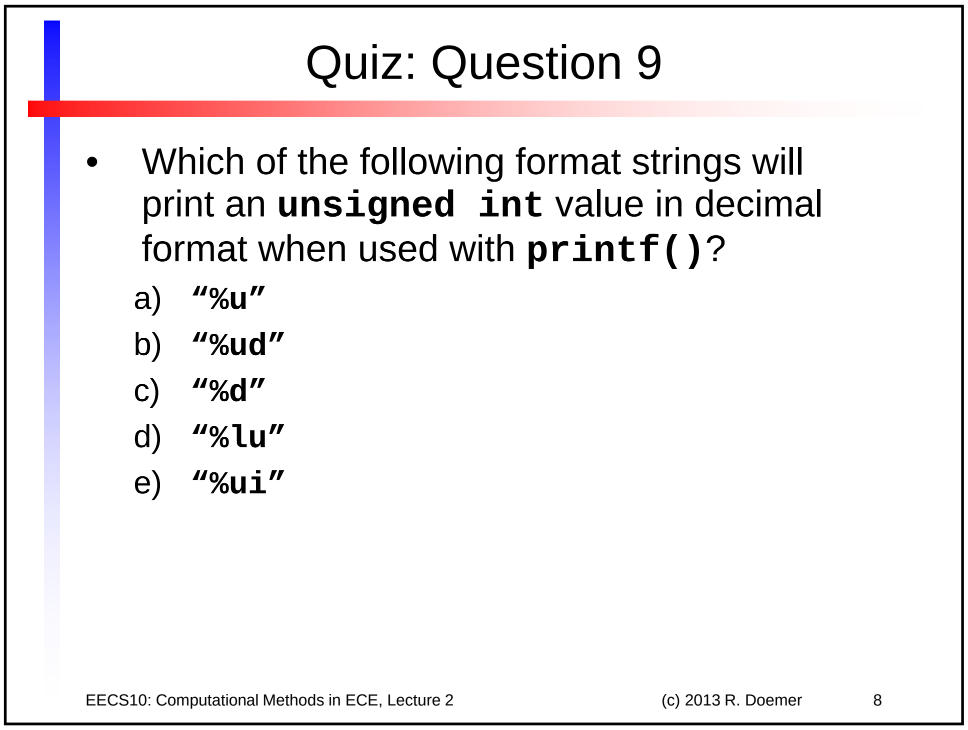


Quiz: Question 8

- What is the value of the integer **`x`** after the following statement?

```
x = (10 – (3 – (20 - -10)));
```

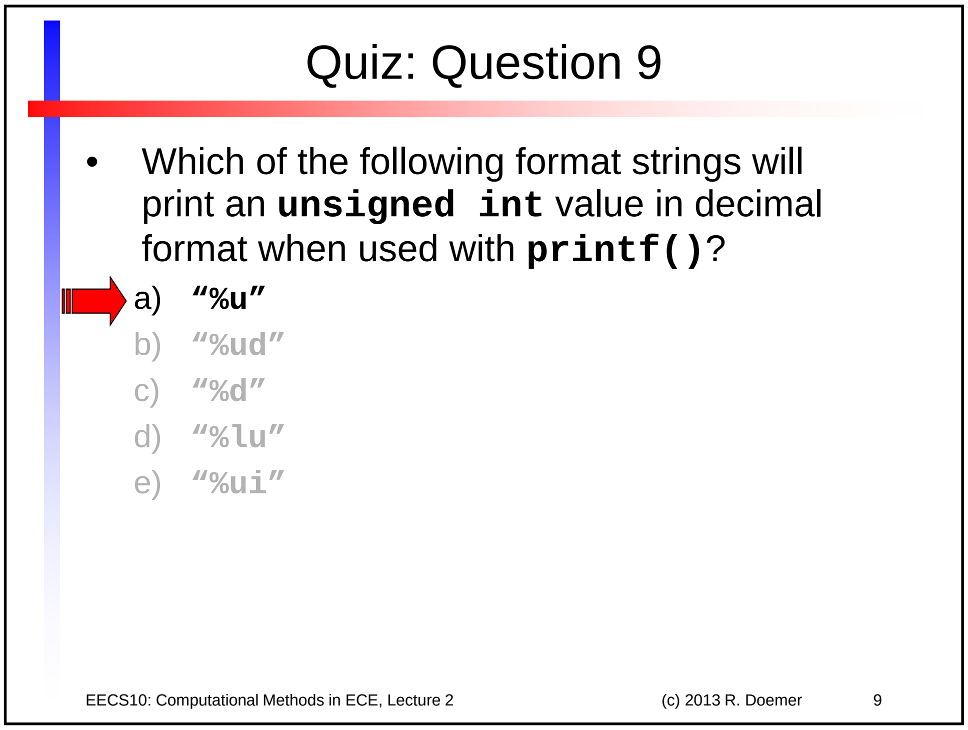
a) **7**
b) **17**
c) **27**
[✓ d) **37**]
e) **77**



Quiz: Question 9

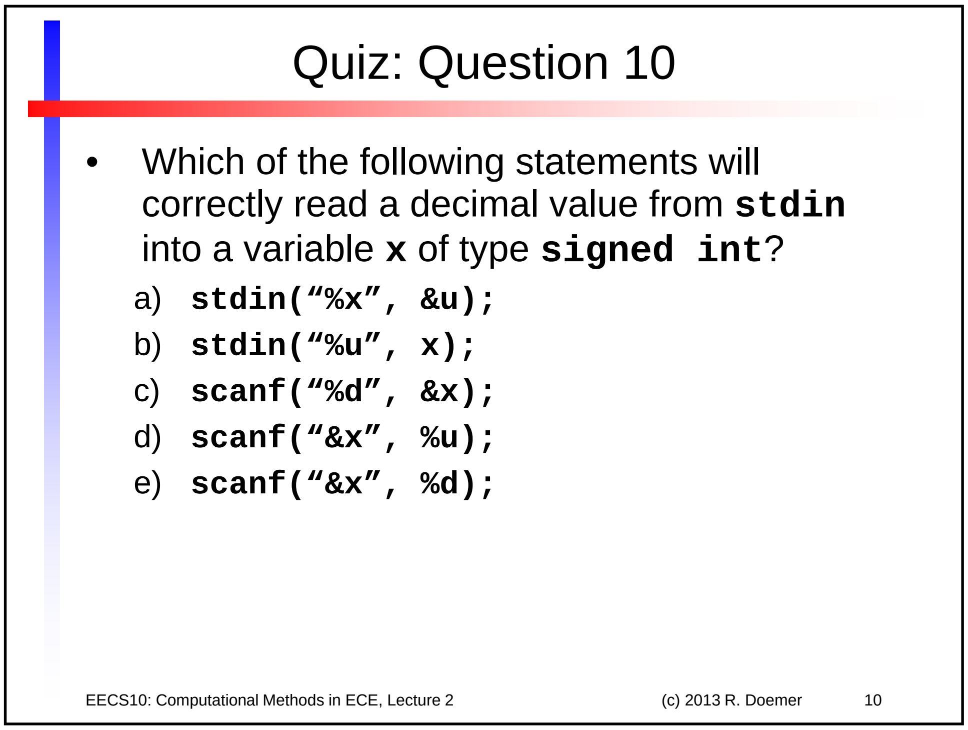
- Which of the following format strings will print an **`unsigned int`** value in decimal format when used with **`printf()`**?

a) **`“%u”`**
b) **`“%ud”`**
c) **`“%d”`**
d) **`“%lu”`**
e) **`“%ui”`**



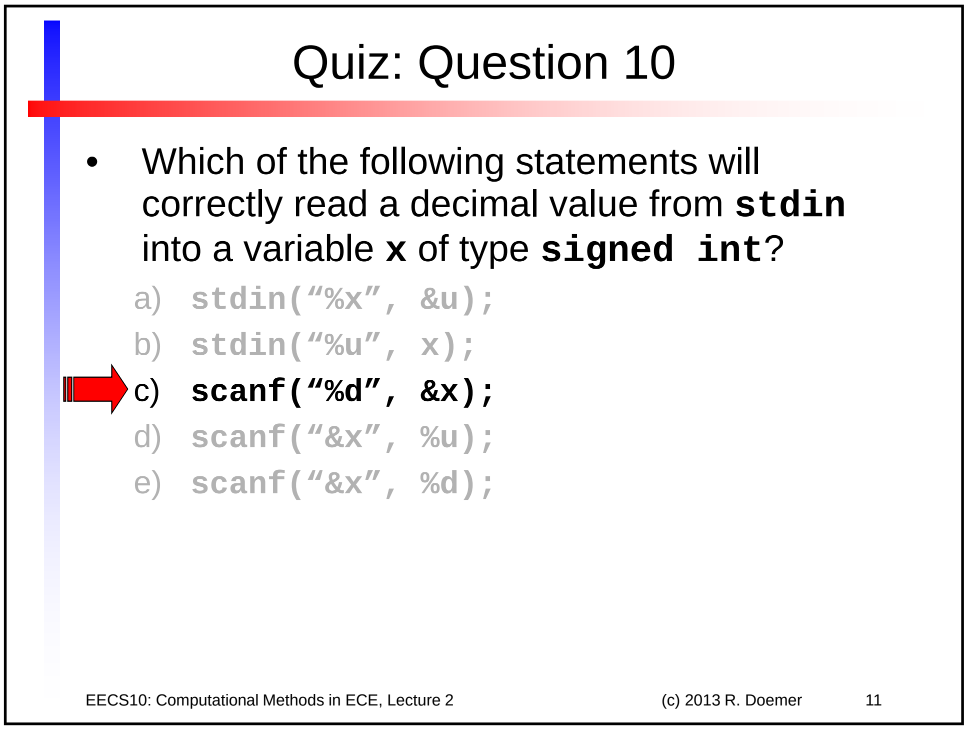
Quiz: Question 9

- Which of the following format strings will print an **`unsigned int`** value in decimal format when used with **`printf()`**?
  - [✓ a) **“%u”**]
  - b) **“%ud”**
  - c) **“%d”**
  - d) **“%lu”**
  - e) **“%ui”**



Quiz: Question 10

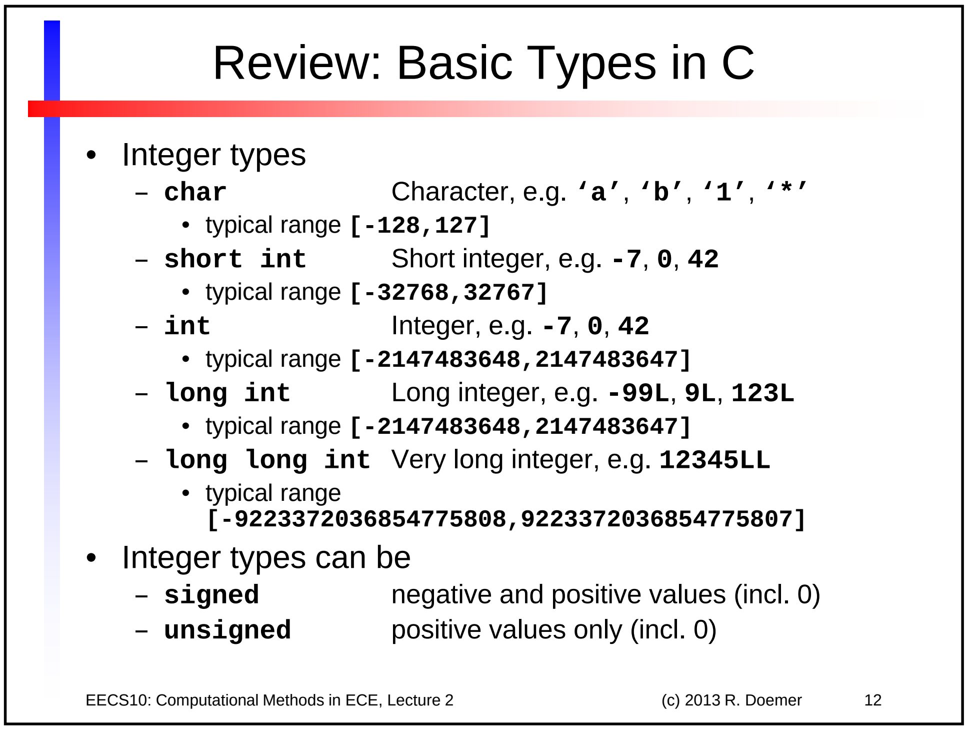
- Which of the following statements will correctly read a decimal value from **`stdin`** into a variable **`x`** of type **`signed int`**?
  - a) **`stdin(“%x”, &u);`**
  - b) **`stdin(“%u”, x);`**
  - c) **`scanf(“%d”, &x);`**
  - d) **`scanf(“&x”, %u);`**
  - e) **`scanf(“&x”, %d);`**



Quiz: Question 10

- Which of the following statements will correctly read a decimal value from **`stdin`** into a variable **`x`** of type **`signed int`**?
  - a) `stdin("%x", &u);`
  - b) `stdin("%u", x);`
  - [✓ c)] **`scanf("%d", &x);`**
  - d) `scanf("&x", %u);`
  - e) `scanf("&x", %d);`

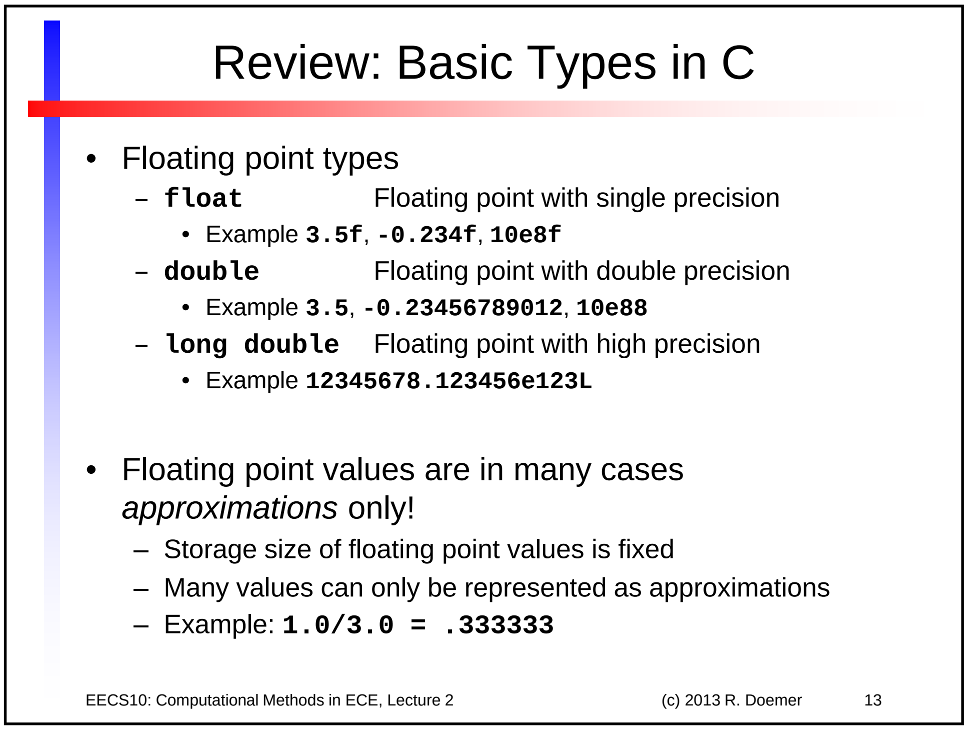
EECS10: Computational Methods in ECE, Lecture 2 (c) 2013 R. Doemer 11



Review: Basic Types in C

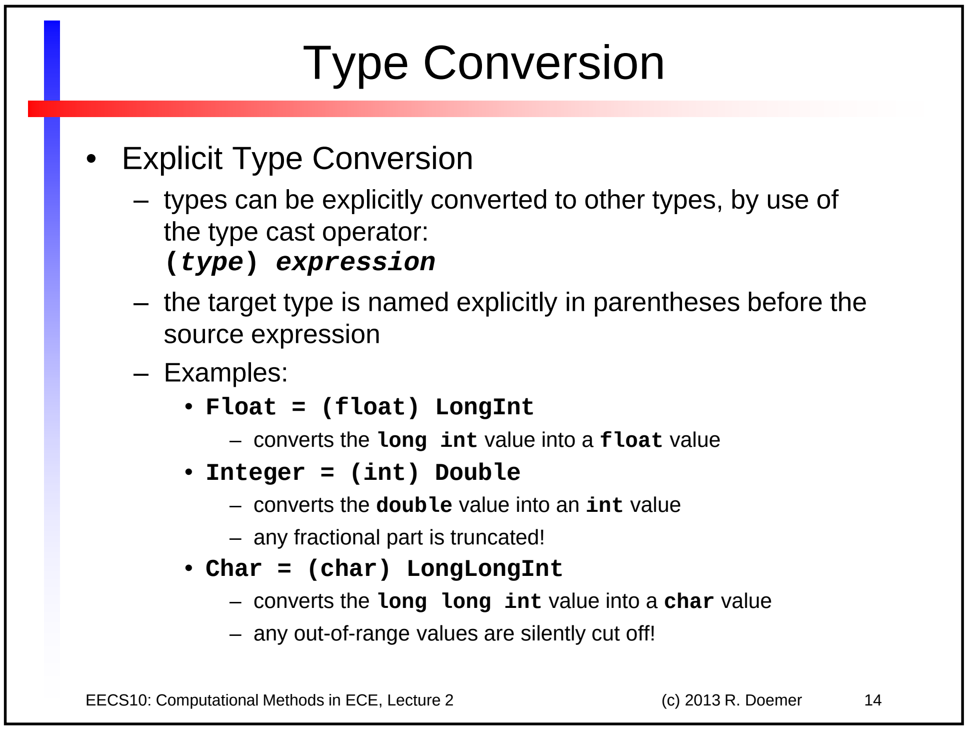
- Integer types
  - **`char`** Character, e.g. **`'a'`**, **`'b'`**, **`'1'`**, **`'*'`**
    - typical range **`[-128,127]`**
  - **`short int`** Short integer, e.g. **`-7`**, **`0`**, **`42`**
    - typical range **`[-32768,32767]`**
  - **`int`** Integer, e.g. **`-7`**, **`0`**, **`42`**
    - typical range **`[-2147483648,2147483647]`**
  - **`long int`** Long integer, e.g. **`-99L`**, **`9L`**, **`123L`**
    - typical range **`[-2147483648,2147483647]`**
  - **`long long int`** Very long integer, e.g. **`12345LL`**
    - typical range **`[-9223372036854775808,9223372036854775807]`**
- Integer types can be
  - **`signed`** negative and positive values (incl. 0)
  - **`unsigned`** positive values only (incl. 0)

EECS10: Computational Methods in ECE, Lecture 2 (c) 2013 R. Doemer 12



Review: Basic Types in C

- Floating point types
  - **`float`** Floating point with single precision
    - Example **`3.5f`**, **`-0.234f`**, **`10e8f`**
  - **`double`** Floating point with double precision
    - Example **`3.5`**, **`-0.23456789012`**, **`10e88`**
  - **`long double`** Floating point with high precision
    - Example **`12345678.123456e123L`**
- Floating point values are in many cases *approximations* only!
  - Storage size of floating point values is fixed
  - Many values can only be represented as approximations
  - Example: **`1.0/3.0 = .333333`**



Type Conversion

- Explicit Type Conversion
  - types can be explicitly converted to other types, by use of the type cast operator:
    `(`*`type`*`)` ***`expression`***
  - the target type is named explicitly in parentheses before the source expression
  - Examples:
    - **`Float = (float) LongInt`**
      - converts the **`long int`** value into a **`float`** value
    - **`Integer = (int) Double`**
      - converts the **`double`** value into an **`int`** value
      - any fractional part is truncated!
    - **`Char = (char) LongLongInt`**
      - converts the **`long long int`** value into a **`char`** value
      - any out-of-range values are silently cut off!

Type Conversion

- Implicit Type Conversion
  - Type promotion
    - integral promotion
      - **`unsigned`** or **`signed char`** is promoted to **`unsigned`** or **`signed int`** before any operation
      - **`unsigned`** or **`signed short`** is promoted to **`unsigned`** or **`signed int`** before any operation
    - binary arithmetic operators are defined only for same types
      - the smaller type is converted to the larger type (before operation)
      - Examples:
        - » **`ShortInt * LongInt`** results in a **`long int`** type
        - » **`LongDouble * Float`** results in a **`long double`** type
  - Type coercion
    - most types are automatically converted to expected types
    - Example: **`Double = Float`**, or **`Char = LongInt`**

Types in Expressions

- Expressions are composed of constants, variables and operators, each of which has an associated type
- Example:

```
short int s;
int       i;
long int  l;
float     f;
double    d;

l  =   2 * s   +   i * f   -   0.5   * d;
```

$$\underbrace{\underbrace{\texttt{l}}_{\text{long}} = \underbrace{\underbrace{\underbrace{\underbrace{\texttt{2}}_{\text{int}} * \underbrace{\texttt{s}}_{\text{int}}}_{\text{int}} + \underbrace{\underbrace{\texttt{i}}_{\text{int}} * \underbrace{\texttt{f}}_{\text{float}}}_{\text{float}}}_{\text{float}} - \underbrace{\underbrace{\texttt{0.5}}_{\text{double}} * \underbrace{\texttt{d}}_{\text{double}}}_{\text{double}}}_{\text{double}}}_{\text{long}}$$

Example Program

- Program example:
  - Task: Write a C program that exercises arithmetic computation by use of different types and operators!
  - The program should compute the following equations:
    - Polynomial:

$$p = 2x^2 - 3x + 5$$

    - Quotient of sums:

$$q = \frac{a + b}{c + d}$$

    - Remainder:

$$r = rem(2^n / 7)$$

  - Assume that a, b, c, d, and n are whole numbers.

Example Program

- Program example: **`Arithmetic.c`** (part 1/3)

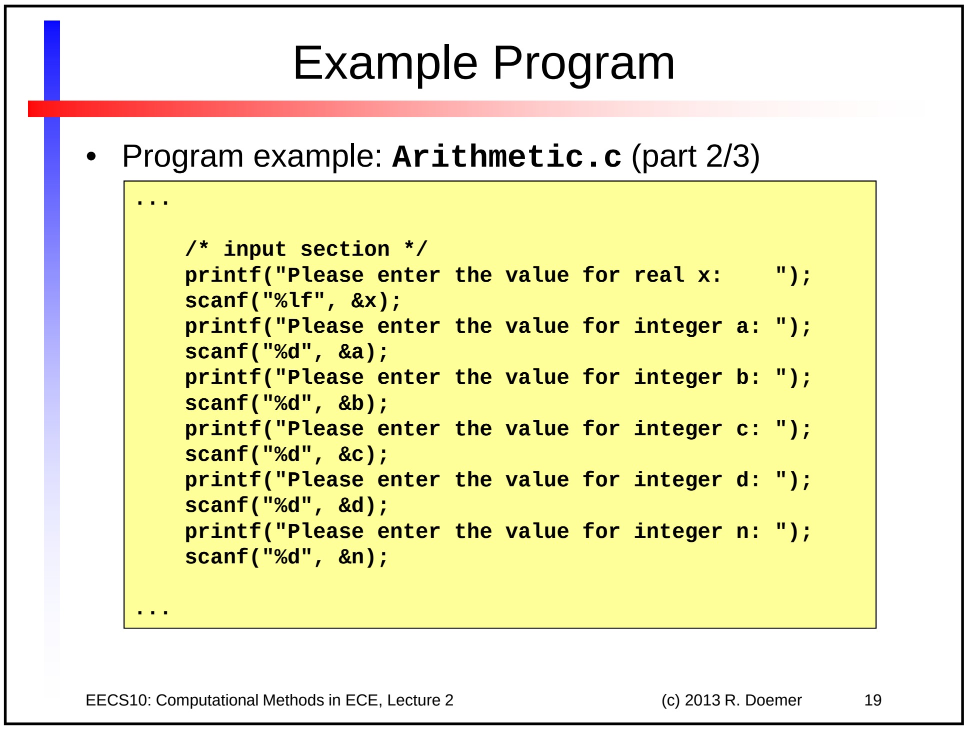
```
/* Arithmetic.c: arithmetic expresions          */
/*                                              */
/* author: Rainer Doemer                        */
/*                                              */
/* modifications:                               */
/* 10/06/04 RD  initial version                 */

#include <stdio.h>

/* main function */

int main(void)
{
    /* variable definitions */
    int    a, b, c, d, n;
    double p, q, r, x;

...
```



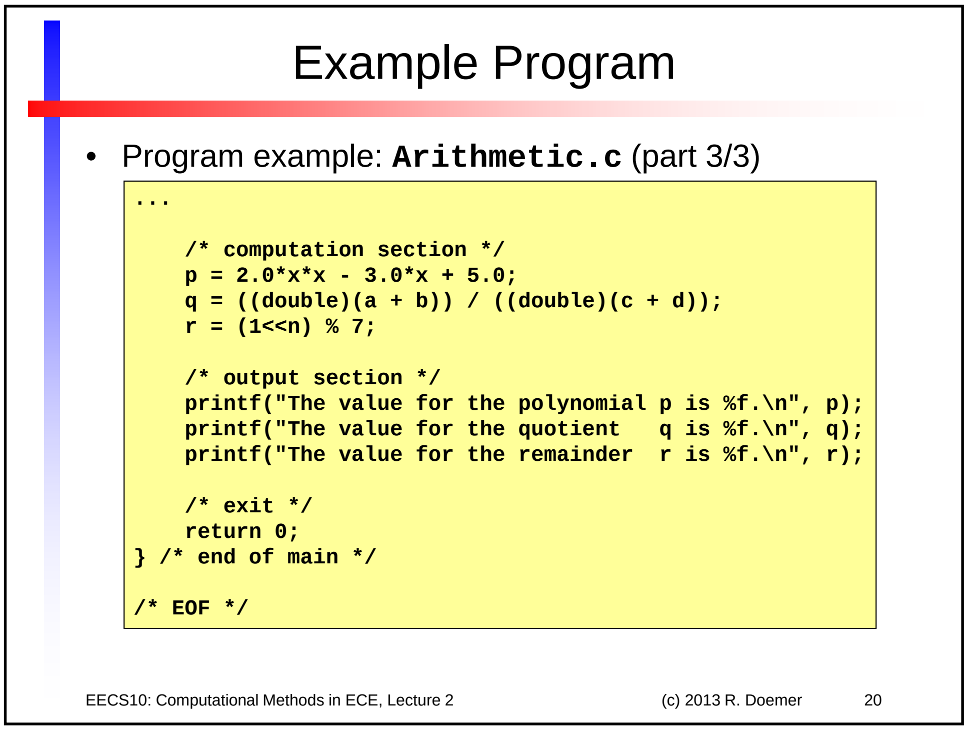
Example Program

- Program example: **`Arithmetic.c`** (part 2/3)

```
...

    /* input section */
    printf("Please enter the value for real x:    ");
    scanf("%lf", &x);
    printf("Please enter the value for integer a: ");
    scanf("%d", &a);
    printf("Please enter the value for integer b: ");
    scanf("%d", &b);
    printf("Please enter the value for integer c: ");
    scanf("%d", &c);
    printf("Please enter the value for integer d: ");
    scanf("%d", &d);
    printf("Please enter the value for integer n: ");
    scanf("%d", &n);

...
```



Example Program

- Program example: **`Arithmetic.c`** (part 3/3)

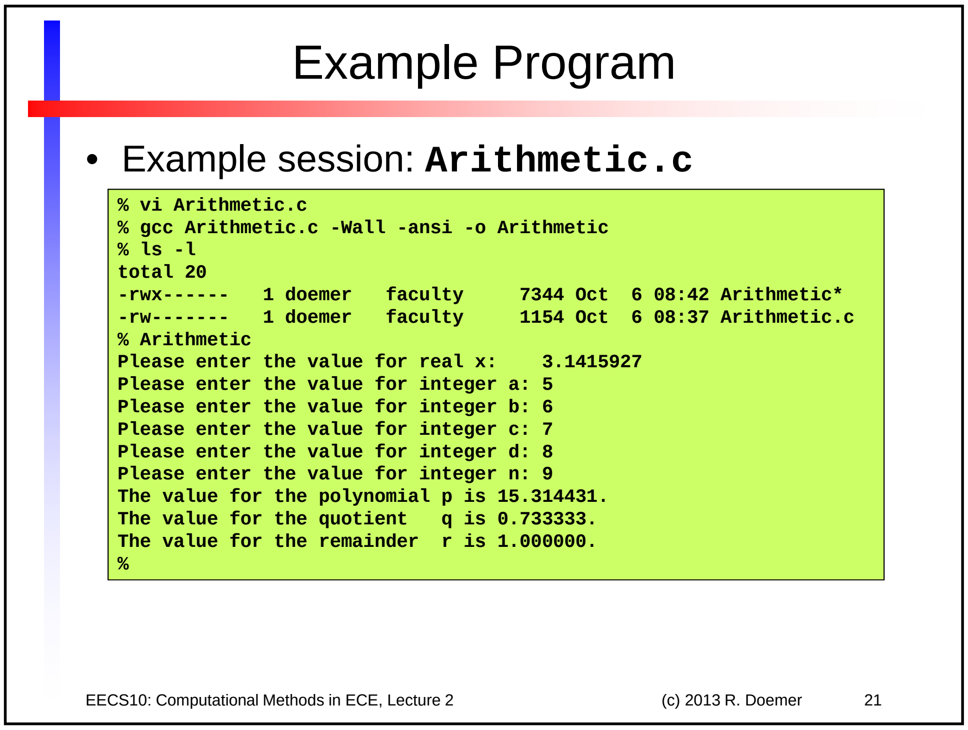
```
...

    /* computation section */
    p = 2.0*x*x - 3.0*x + 5.0;
    q = ((double)(a + b)) / ((double)(c + d));
    r = (1<<n) % 7;

    /* output section */
    printf("The value for the polynomial p is %f.\n", p);
    printf("The value for the quotient   q is %f.\n", q);
    printf("The value for the remainder  r is %f.\n", r);

    /* exit */
    return 0;
} /* end of main */

/* EOF */
```



Example Program

- Example session: **`Arithmetic.c`**

```
% vi Arithmetic.c
% gcc Arithmetic.c -Wall -ansi -o Arithmetic
% ls -l
total 20
-rwx------   1 doemer   faculty     7344 Oct  6 08:42 Arithmetic*
-rw-------   1 doemer   faculty     1154 Oct  6 08:37 Arithmetic.c
% Arithmetic
Please enter the value for real x:    3.1415927
Please enter the value for integer a: 5
Please enter the value for integer b: 6
Please enter the value for integer c: 7
Please enter the value for integer d: 8
Please enter the value for integer n: 9
The value for the polynomial p is 15.314431.
The value for the quotient   q is 0.733333.
The value for the remainder  r is 1.000000.
%
```