More C

9/16/15
Which of the following lines will print the value of an int x?

A. printf(x);
B. printf(“%d\n”, x);
C. printf(“%i\n”, x);
D. printf(“%f\n”, x);
E. What’s an int?
Which of the following lines will print the value of an int x?

A. printf(x);
B. printf(“%d\n”, x);
C. printf(“%i\n”, x);
D. printf(“%f\n”, x);
E. What’s an int?
Which of the following is different between Java and C (99)?

A. The format of comments
B. The way variables are declared
C. The syntax of while loops
D. The way indentation and code blocks (within braces) are treated
E. The way it is compiled and run
Which of the following is different between Java and C (99)?

A. The format of comments
B. The way variables are declared
C. The syntax of while loops (sort of)
D. The way indentation and code blocks (within braces) are treated
E. The way it is compiled and run
Similarities between C and Java
Difference: No boolean type

• Uses integers instead
  
  0 = false
  
  Anything else = true
Difference: 1-pass compilation

• Must declare functions before they are used (or compiler will guess at the type...)

```c
void myFun(); //declaration = signature w/o body

//calls to the function work properly here

void myFun() { //actual definition of the function ...
```
The program below wants to read an integer into the variable x. Which of the following lines is incorrect?

```
#include <stdio.h>    //A

int main() {         //B
    int x = 0;        //C
    scanf("%d\n", x); //D
    ...                //E
```
The program below wants to read an integer into the variable x. Which of the following lines is incorrect?

```
#include <stdio.h>  //A
int main() {      //B
  int x = 0;       //C
  scanf("%d\n", x); //D
...               //E
```

Why the ampersand?

• New operator in C

• Gets the memory address of a variable
  – scanf needs to put its value somewhere
Pointers

• Special kind of variable that stores a memory address

```c
int x;       // x stores an int
int* p;      // p is a pointer to an integer
              // type is pronounced "int star"

p = &x;      // set using the ampersand

*p = 5;      // follow the pointer using * to access or
              // change the value to which it points
```
Which of the following is **NOT** true about pointers?

A. They are variables that store memory addresses
B. Their value can be set using the & operator on another variable
C. Their declarations include *
D. The * operator is used to change their value
E. All of the above are true
Which of the following is **NOT** true about pointers?

A. They are variables that store memory addresses
B. Their value can be set using the & operator on another variable
C. Their declarations include *
D. The * operator is used to change their value
E. All of the above are true
More about pointers

• Special value NULL (generally 0)
• Allowed to have pointers to pointers:
  ```c
  int x;
  int* p = &x;
  int** p2 = &p;
  ```
• Can assign pointers (copy addresses):
  ```c
  int* ptr1 = ...;
  int* ptr2 = ptr1;
  ```