The instruction cycle is executed by

(a) ALU of CPU
(b) RAM
(c) disk drive
(d) monitor
(e) control unit of CPU
RAM refers to

(a) disk drive
(b) main memory
(c) monitor
(d) CPU
(e) printer
Which of the following is not a valid C identifier?

(a) int
(b) _int
(c) int4
(d) int_4
(e) _4
Which of the following is a placeholder for an int value in an I/O format string?

(a) &f
(b) &d
(c) %f
(d) %d
(e) &lf
Which is the remainder of 9 divided by 5?

(a) 9 / 5
(b) 9 // 5
(c) 9 & 5
(d) 9 % 5
(e) 9 @ 5
What does
    int x = 3; print("%d", ++x); printf(" %d", x); print?

(a) 3 3
(b) 3 4
(c) 3 5
(d) 4 4
(e) 4 5
What does
   `int x = 7; x += x + 1; printf(" %d", x);`
print?

(a) 7
(b) 8
(c) 9
(d) 14
(e) 15
What does
   int x = 3; x += 1; printf(" %d", x);
print?

(a) 1
(b) 2
(c) 3
(d) 4
(e) 5
Review

* variables
* data types: int, double, char, bool(, float, ...)
  7 vs. 7. vs. ‘7’ vs. true (vs. 7.f)
* operators
  a + b, a - b, a * b, a / b, a % b,
  - a, + a
  (int), (double), (char), (float) can be implicit
if you #include <stdbool.h>
  (bool)
if you #include <math.h>
  pow(a, b), sqrt(a)
Which of the following is undefined in C? (assuming that int x = 3; int y = 4, say.)

(a) x += 1  
(b) x = x++  
(c) x = y++  
(d) x = y = 5  
(e) x = ++y
What is 3.f/2?

(a) 1
(b) undefined
(c) 1.5
(d) 1.0f
(e) 1.5f
To print a number to two decimal places right justified in a field of width 10, use the format string

(a) “%7.2d”
(b) “%10.2d”
(c) “%8.2d”
(d) “%10.2f”
(e) “%7.2f”
Note.

Since the advent of the IBM 360 in the 1965, it has become customary to use double precision as the default for floating-point values ("real" in the textbook). Though the textbook uses double precision (double) constants, it declares variables to be single precision (float). The assignments in this course follow the book.
Why use functions?

- Problems can be factored into understandable and manageable sub-problems.
- Functions provide a way to reuse code in our program.
- Functions provide a way to reuse code in programs that have yet to be developed.
- Functions assist us in protecting the data in the calling function (e.g., in `main`).
Declare vs. Define

definition = declaration + storage allocation
(the textbook *sometimes* suggests definition = storage allocation)

- every function (and variable) must be defined exactly once
- every function (and variable) must be declared before use
- a function can be declared more than once
  (there is no way to declare a local variable apart from defining it)
Basic Function Designs

Functions are of two types:
• those that return no value
• those that return one value
There is a difference in syntax.

a function may have 0, 1, 2, ... arguments
int integerDivision(int x, int y)
{
    return x / y;
}

int calculateRemainder(int x, int y)
{
    int remain;  // WILL HOLD THE RESULT
    remain = x % y;
    return remain;
}

How is the variable x in integerDivision related to the variable x in calculateRemainder?
Local variables in different functions having the same name are different variables.
Parameter Passing

The way that C passes parameters is simple and consistent though not always intuitive:

- The evaluated arguments from the calling program are assigned to the parameters of the called program. (These parameters are distinct variables local to the called program.)

- After the return, these parameters and their values vanish.
swap example --->

Pass by Value (downward communication) versus Pass by Address (upward communication)

textbook: “The C language uses only pass by value ...”

(that value can be an address)
Which prototype is correct for a function that returns no value?

(a) int welcome(void);
(b) int welcome();
(c) welcome();
(d) void welcome();
(e) welcome(void);
int a = 3;
increment(a); printf("%d ", a);
...
void increment(int x)
{
    x = x + 1; printf("%d ", x);
}
What is printed?  Careful.

(a) 3 3
(b) 3 4
(c) 4 3
(d) 4 4
int x = 1; int y = 2; // !@#
display(y);
...
void display(int z){
    int x; int y;
    printf("%d %d %d", x, y, z);
}
Which values in printf are well defined?
(a) z only
(b) y, z only
(c) x, y only
(d) y only
(e) x, y, z