**5.1 Introduction & 5.2 Modularizing Programs in C**

5.1 All of the following are *true* of functions except:

(a) they define specific tasks that can be used at many points in a program

(b) a function call must specify the name and arguments of the function

(c) the definition of a function is always visible to other functions

(d) the implementation of a function is hidden from the caller

ANS: (c)

5.2 Experience has shown that the best way to construct a program is from small pieces. This is called \_\_\_\_\_\_\_\_\_\_.

a) bottom up

b) the whole is greater than the sum of the parts

c) divide and conquer

d) recursion

ANS: (c)

5.3 Which is *not* an ANSI standard library function?

a) printf

b) main

c) scanf

d) pow

ANS: (b)

5.4 Which one item is most *different* from the other three?

a) worker function

b) caller

c) calling function

d) boss function

ANS: (a)

5.5 When a called function completes its task, it normally

a) terminates program execution normally

b) aborts program execution

c) logs its results

d) returns to the calling function

ANS: (d)

5.6 Which statement is *true*?

a) The boss function normally knows how the worker function performs its designated tasks.

b) A worker function may not call other worker functions.

c) “Hiding” of implementation details makes it difficult to understand software.

d) The boss function is normally unaware when a worker function calls another function.

ANS: (d)

5.7 Functions are \_\_\_\_\_\_\_\_\_\_ by a function call.

a) inveigled

b) invoked

c) internalized

d) inverted

ANS: (b)

**5.3 Math Library Functions**

5.8 All functions in the math library return the data type \_\_\_\_\_\_\_\_\_\_.

a) float

b) int

c) long

d) double

ANS: (d)

5.9 If a = 7.0, b = 7.0 and c = 6.0, then what is printed by

printf("%.2f", sqrt(a + b \* c));

a) 49

b) 7.00

c) 7

d) 49.00

ANS: (b)

5.10 What is the value of fabs(-5.0)?

a) 5

b) 5.0

c) -5

d) –5.0

ANS: (b)

5.11 Which of the following is *not* included in <math.h>?

(a) pow

(b) floor

(c) ln

(d) log10

ANS: (c)

**5.4 Functions**

5.12 A *valid* reason for building programs out of functions is

(a) that the divide-and-conquer approach facilitates program construction

(b) that pre-existing functions can be used to create new programs

(c) the avoidance of code iteration within a program

(d) all of the above

ANS: (d)

5.13 Which is *not* a motivation for “functionalizing” a program?

a) The divide-and-conquer approach makes program development more manageable.

b) Software reusability—using existing building blocks to create new programs.

c) Avoid repeating code.

d) Execution performance—functionalized programs run faster.

ANS: (d)

5.14 Which statement is *false*?

a) Each function should be limited to performing a single, well-defined task.

b) If you cannot choose a concise name that expresses what a function does, it’s possible that the function is attempting to perform too many diverse tasks.

c) Every function should be broken into smaller functions.

d) A function’s parameters are local variables.

ANS: (c)

**5.5 Function Definitions**

5.15 The function prototype

double mySqrt(int x);

(a) defines a function called mySqrt which takes an integer as an argument and returns a double

(b) defines a function called double which calculates square roots

(c) defines a function called mySqrt which takes an argument of type x and returns a double

(d) defines a function called mySqrt which takes a double as an argument and returns an integer

ANS: (a)

5.16 Which of the following will *not* produce a syntax error?

(a) Omitting a return type from a function definition if the function prototype specifies a return type other than int

(b) Returning a value from a function defined as void

(c) Defining a function parameter again inside a function

(d) Using the same names for arguments passed to a function and the corresponding parameters in the function definition

ANS: (d)

5.17 Which of the following is *not* an indication that a function may be too complex?

(a) it has a large size

(b) it has a large parameter list

(c) its name is a clear reflection of its function

(d) it performs multiple tasks

ANS: (c)

5.18 Which of the following code segments does *not* contain any errors?

(a)   
void printnum (int x)

{

print("%i", x);

return x;

}

(b)   
int cube(int s)

{

int s;

return (s \* s \* s);

}

(c)   
double triple(float n) {

return (3 \* n);  
}

(d)   
double circumference (int r)

return (3.14 \* 2 \* r);

ANS: (c)

5.19 int square(int); is an example of a function \_\_\_\_\_\_\_\_\_\_.

a) datatype

b) stereotype

c) prototype

d) proceduretype

ANS: (c)

5.20 As used in

int square(int);

int is not a(n) \_\_\_\_\_\_\_\_\_\_.

a) data type

b) parameter type

c) return type

d) function prototype

ANS: (d)

5.21 The type of a parameter whose type is omitted in a function definition is \_\_\_\_\_\_\_\_\_\_.

a) int

b) double

c) long

d) float

ANS: (a)

5.22 The most concise notation to use to define function parameters x and y as double is \_\_\_\_\_\_\_\_\_\_.

a) x, y

b) x, double y

c) double x, y

d) double x, double y

ANS: (d)

5.23 Placing a semicolon after the right parenthesis enclosing the parameter list of a function definition is a \_\_\_\_\_\_\_\_\_\_ error.

a) logic

b) syntax

c) fatal runtime

d) nonfatal runtime

ANS: (b)

5.24 Which statement is *true*?

a) The type of every parameter in a function parameter list must be included.

b) The type of every argument in a function call must be included.

c) It is not incorrect to use the same names for the arguments passed to a function and the corresponding parameters in the function definition.

d) Defining a function parameter again as a local variable within the function is a logic error.

ANS: (c)

5.25 Which statement is *false*?

a) Every block is a compound statement.

b) Every compound statement is a block.

c) Blocks can be nested.

d) Compound statements can be nested.

ANS: (b)

5.26 Which statement is *true*?

a) Programs should be written as collections of small functions.

b) A function must be no longer than one page.

c) The best engineered functions have many parameters and perform many distinct tasks.

d) Every function must include an explicit return statement.

ANS: (a)

**5.6 Function Prototypes: A Deeper Look**

5.27 A function prototype does *not* have to \_\_\_\_\_\_\_\_.

(a) include parameter names

(b) terminate with a semicolon

(c) agree with the function definition

(d) match with all calls to the function

ANS: (a)

5.28 A function prototype can always be omitted when a function \_\_\_\_\_\_\_\_.

(a) is defined before it is first invoked

(b) is invoked before it is first defined

(c) takes no arguments

(d) does not return a value

ANS: (a)

5.29 Which statement is *false*?

a) The compiler uses function prototypes to validate function calls.

b) Prior to ANSI C, C did not include function prototypes.

c) A function prototype tells the compiler the type of data returned by the function, the number of parameters the function expects to receive, the types of these parameters and the order in which parameters of these types are expected.

d) The designers of ANSI C++ borrowed the notion of function prototypes from the developers of C.

ANS: (d)

5.30 The forcing of arguments to the appropriate types is commonly called \_\_\_\_\_\_\_\_\_\_.

a) conversion

b) casting

c) coercion

d) transmogrification

ANS: (c)

**5.7 Function Call Stack and Stack Frames**

No questions.

**5.8 Header Files**

5.31 Each standard library has a corresponding \_\_\_\_\_\_\_\_\_\_.

(a) function

(b) variable type

(c) header file

(d) cd-rom

ANS: (c)

5.32 Which standard library header file contains function prototypes for conversions of numbers to text and text to numbers, memory allocation, random numbers and other utility functions.

a) <stdarg.h>

b) <stdlib.h>

c) <stdutl.h>

d) <stddef.h>

ANS: (b)

**5.9 Passing Arguments By Value and By Reference**

5.33 When arguments are passed by \_\_\_\_\_\_\_\_\_\_, the caller allows the called function to modify the original variable’s value.

(a) value

(b) reference

(c) both a and b

(d) none of these

ANS: (b)

5.34 Which statement is *true*?

a) When an argument is passed call by reference, a copy of the argument’s value is made and passed to the called function.

b) With call by reference, changes to the passed value do not affect the original variable’s value in the calling functions.

c) Call by value should be used whenever the called function does not need to modify the value of the caller’s original value.

d) Call by value should only be used with trusted called functions that need to modify the original variable.

ANS: (c)

**5.10 Random Number Generation**

5.35 The randfunction generates a data value of the type

(a) unsigned int

(b) int

(c) long int

(d) short int

ANS: (b)

5.36 A variable that can have values only in the range 0 to 65535 is a

(a) four-byte int

(b) four-byte unsigned int

(c) two-byte int

(d) two-byte unsigned int

ANS: (d)

5.37 In the expression

n = a + rand() % b;

(a) b is the shifting value

(b) a is the scaling value

(c) b is equal to the width of the desired range of integers

(d) both (a) and (c)

ANS: (c)

5.38 srand  
(a) should be called before each call to rand

(b) should be used instead of rand to generate truly random numbers

(c) is unnecessary in C

(d) can use timeas an automatically input seed value

ANS: (d)

5.39 Which statement is *false*?

a) Function rand generates an integer between 0 and MAX.

b) The range of values produced directly by rand is often different than what is needed in a specific application.

c) The number 6 in the expression rand % 6 is called a scaling factor.

d) The rand function prototype is in <stdlib.h>.

ANS: (a)

**5.11 Example: A Game of Chance**

No questions.

**5.12 Storage Classes**

5.40 An identifier’s storage class

(a) determines the period during which that identifier exists in memory

(b) determines whether an identifier in a multiple-source-file program is known only in the current source file or in any source file with proper definitions

(c) determines where the identifier can be referenced in a program

(d) all of the above

ANS: (d)

5.41 Which of the following is *not* true of static local variables?

(a) They’re accessible outside of the function in which they are defined.

(b) They retain their values when the function is exited.

(c) They’re initialized to zero if not explicitly initialized by the programmer.

(d) They can be pointers.

ANS: (a)

5.42 Which is *not* an attribute of a variable?

a) name

b) definition

c) type

d) value

ANS: (b)

5.43 Which is *not* a storage class?

a) automatic

b) register

c) extern

d) static

ANS: (a)

5.44 Which is *not* an attribute of a variable?

a) storage class

b) storage duration

c) scope

d) external class

ANS: (d)

5.45 An identifier’s \_\_\_\_\_\_\_\_\_\_ is where the identifier can be referenced in a program.

a) locality

b) vicinity

c) neighborhood

d) scope

ANS: (d)

5.48 Global variables and function names are of storage class \_\_\_\_\_\_\_\_\_\_ by default.

a) register

b) extern

c) static

d) auto

ANS: (b)

**5.13 Scope Rules**

5.46 Labels are the *only* identifiers with

(a) function scope

(b) file scope

(c) block scope

(d) function-prototype scope

ANS: (a)

5.47 The only identifiers that can be reused elsewhere in a program without any ambiguity are

(a) global variables

(b) static local variables

(c) those in the parameter list of a function prototype

(d) those in the parameter list of a function definition

ANS: (c)

5.48 Which statement is *false*?

a) When we define a local variable in a block it can be referenced only in that block or in blocks in which that block is nested.

b) Labels are the only identifiers with function scope.

c) Labels can be used anywhere in the function in which they appear, but can not be referenced outside the function body.

d) Labels are used in switch statements and in goto statements.

ANS: (a)

5.49 Which is *not* a scope for an identifier?

a) function scope

b) record scope

c) block scope

d) function-prototype scope

ANS: (b)

**5.14 Recursion**

5.50 A recursive function is a function that \_\_\_\_\_\_\_\_.

(a) returns a double

(b) takes 3 arguments

(c) calls itself

(d) is inside of another function

ANS: (c)

5.51 What value does function mystery return when called with a value of 4?

int mystery (int number)   
{

if (number <= 1) {

return 1;  
 }

else {

return number \* mystery(number – 1);  
 }

}

(a) 1

(b) 24

(c) 0

(d) 4

ANS: (b)

5.52 Recursion is memory-intensive because \_\_\_\_\_\_\_\_.

(a) it must occur numerous times before it terminates

(b) previous function calls are still open when the function calls itself and the arguments of these previous calls still occupy space on the call stack

(c) many copies of the function code are created

(d) it requires large data values

ANS: (b)

5.53 Which statement is *false*?

a) A recursive function is a function that calls itself either directly or indirectly through another function.

b) A recursive function knows how to solve only one or more base cases.

c) The recursion step executes after the original call to the function terminates.

d) In order for the recursion to eventually terminate, each time the function calls itself with a slightly simpler version of the original problem, this sequence of smaller and smaller problems must eventually converge on a base case.

ANS: (c)

**5.15 Example Using Recursion: Fibonacci Series**

5.54 Assuming the following pseudocode for the Fibonacci series, what is the value of fibonacci(5)?

*fibonacci(0) = 0*

*fibonacci(1) = 1*

*fibonacci(n) = fibonacci(n – 1) + fibonacci(n – 2)*

(a) 8

(b) 1

(c) 3

(d) 5

ANS: (d)

5.55 Which statement is *false*?

a) The ANSI C standard does not specify the order in which the operands of most operators are to be evaluated.

b) The ANSI C standard specifies the order of evaluation of operators &&, ||, comma, and ?:.

c) A program that evaluates Fibonacci numbers recursively achieves high performance because of exponential complexity.

d) Programs that depend on the order of evaluation of the operands of operators other than &&, ||, comma, and ?: can function differently on systems with different compilers.

ANS: (c)

**5.16 Recursion vs. Iteration**

5.56 Recursion is to the base case as iteration is to \_\_\_\_\_\_\_\_?

(a) the counter

(b) an iteration statement

(c) failure of the loop continuation test

(d) a selection statement

ANS: (c)

5.57 All of the following are reasons to use recursion except:

(a) an iterative solution is not apparent

(b) the resulting program is easier to debug

(c) it more naturally mirrors the problem

(d) it maximizes software performance

ANS: (d)

5.58 Which statement is *false*?

a) Both recursion and iteration are based on a control statement.

b) Both iteration and recursion involve iteration.

c) Iteration with sentinel-controlled iteration and recursion each gradually approach termination.

d) Both iteration and recursion can occur infinitely.

ANS: (c)

**5.17 Secure C Programming**

5.59 Which of the following statements is true?

(a) The C standard library does not provide a secure random-number generator.

(b) According to the C standard document’s description of function rand, “There are no guarantees as to the quality of the random sequence produced and some implementations are known to produce sequences with distressingly non-random low-order bits.”

(c) The CERT guideline MSC30-C indicates that implementation-specific random-number generation functions must be used to ensure that the random numbers produced are not predictable

(d)All of the above.

ANS: (d)