**9.1 Introduction**

9.1 Which function does *not* use either the standard input stream or the standard output stream?

a) puts

b) getchar

c) gettime

d) scanf

ANS: (c)

**9.2 Streams**

9.2 Which of the following is *not* a standard C formatted stream?

(a) standard input stream

(b) standard error stream

(c) standard output stream

(d) standard redirection stream

ANS: (d)

9.3 Which statement about the standard streams is *true*?

a) The standard input stream must be connected to the keyboard.

b) The standard output stream must be connected to the screen.

c) The standard error stream is connected to the screen by default.

d) Streams may not be redirected.

ANS: (c)

**9.3 Formatting Output with printf**

9.4 Which of the following does *not* represent a capability of the printf function?

(a) center justification of outputs

(b) left justification of outputs

(c) right justification of outputs

(d) inserting literal characters at precise locations in a line of output

ANS: (a)

9.5 Which is *not* a part of a format control string?

a) conversion specifiers

b) flags

c) field widths

d) printf

ANS: (d)

9.6 Which is *not* a formatting capability of printf?

a) left justification

b) centering

c) right justification

d) aligning a column of numbers so that decimal appoints appear one above the other

ANS: (b)

9.7 Which is *not* a formatting capability of printf?

a) representing unsigned integers in binary format

b) representing unsigned integers in decimal format

c) representing unsigned integers in hexadecimal format

d) representing unsigned integers in octal format

ANS: (a)

9.8 A printf format control string must be enclosed in

a) slashes

b) /\* \*/

c) single quotes

d) double quotes

ANS: (d)

**9.4 Printing Integers**

9.9 Which statement about the i and d integer conversion specifiers is false?

a) They display a signed decimal integer (when used with printf).

b) They are identical when used with printf.

c) They are identical when used with scanf.

d) They display numbers without a decimal point.

ANS: (c)

9.10 Which integer conversion specifier displays an unsigned octal integer?

a) oct

b) OCT

c) o (lowercase letter)

d) O (uppercase letter)

ANS: (c)

9.11 Which integer conversion specifier displays an unsigned decimal integer?

a) u

b) U

c) ud

d) UD

ANS: (a)

9.12 Which integer conversion specifier would display the hexadecimal digit A?

a) h

b) H

c) x

d) X

ANS: (d)

9.13 When used with integer conversion specifiers, the letters h and l (“el”) are called \_\_\_\_\_\_\_\_\_\_.

a) height modifiers

b) length modifiers

c) complements

d) adjustors

ANS: (b)

9.14 Place \_\_\_\_\_\_\_\_\_\_ before any integer conversion specifier to indicate that a short integer is to be displayed.

a) s

b) S

c) h

d) H

ANS: (c)

9.15 Which statement about integer conversion specifiers is *false*?

a) It is an error to print a negative value with a conversion specifier that expects an unsigned value.

b) –455 when printed with %u prints 455.

c) When printing an integers with %d, the plus sign (on a positive integer) does not print.

d) Place an l (“el”) before any integer conversion specifier to indicate that a long integer is displayed.

ANS: (b)

**9.5 Printing Floating-Point Numbers**

9.16 The %e conversion specifier displays \_\_\_\_\_\_\_\_ values.

(a) long

(b) character

(c) integer

(d) floating-point

ANS: (d)

9.17 The %g conversion specifier indicates \_\_\_\_\_\_\_\_.

(a) color

(b) significant digits

(c) a global variable

(d) a hexadecimal integer

ANS: (b)

9.18 A floating-point value always contains a \_\_\_\_\_\_\_\_\_\_.

a) decimal point

b) comma

c) plus sign

d) e or E

ANS: (a)

9.19 The floating-point conversion specifiers e and E display floating-point values in \_\_\_\_\_\_\_\_\_\_ notation.

a) elliptical

b) existential

c) exponential

d) exportable

ANS: (c)

9.20 In 1.504582E+02, the E+02 indicates that \_\_\_\_\_\_\_\_\_\_.

a) 1.504582 is to be raised to the second power

b) 1.504582 is to be extended by two print positions

c) 1.504582 is to be multiplied by 100

d) 1.504582 is to be doubled

ANS: (c)

9.21 Values printed with the conversion specifiers e, E and f are output with \_\_\_\_\_\_\_\_\_\_ digits of precision to the right of the decimal point by default.

a) 0

b) 1

c) 5

d) 6

ANS: (d)

9.22 Conversion specifiers e and E always print exactly \_\_\_\_\_\_\_\_\_\_ to the left of the decimal point.

a) zero digits

b) one digit

c) two digits

d) three digits

ANS: (b)

9.23 Conversion specifiers g or G always print \_\_\_\_\_\_\_\_\_\_.

a) no trailing zeros

b) one trailing zero

c) as many trailing zeros as are in the number itself

d) a default of six trailing zeros

ANS: (a)

9.24 With the %g conversion specifier, the value 0.0000875 prints as

a) 8.75e-05

b) 87.5e-06

c) 0.875e-04

d) 0 (because of truncation)

ANS: (a)

**9.6 Printing Strings and Characters**

9.25 What would be the output of the following statements?

char\* value = "hello";

printf( "%c", value );

(a) h

(b) hello

(c) value

(d) none of these

ANS: (d)

9.26 What would be the output of the following statement?

char\* value = 'hello';

printf( '%s', value );

(a) h

(b) hello

(c) value

(d) none of these

ANS: (d)

9.27 Conversion specifier c requires a(n) \_\_\_\_\_\_\_\_\_\_ argument.

a) pointer to char

b) char

c) integer

d) ASCII numeric

ANS: (b)

9.28 Conversion specifier s requires a(n) \_\_\_\_\_\_\_\_\_\_ argument.

a) pointer to char

b) char

c) integer

d) ASCII numeric

ANS: (a)

9.29 Which is *not* a common programming error with regard to the character and string conversion specifiers.

a) Using %c to print a string.

b) Using %s to print a char argument

c) Using double quotes around a character string

d) Using double quotes around a character constant

ANS: (c)

**9.7 Other Conversion Specifiers**

9.30 Which of the following would output a “%” sign when used in the format control string of the printf statement?

(a) %/

(b) /%

(c) %%

(d) none of these

ANS: (c)

9.31 Conversion specifier p \_\_\_\_\_\_\_\_\_\_.

a) displays a pointer value with -> notation

b) displays a pointer value in hexadecimal notation

c) displays a pointer value in decimal notation

d) displays a pointer value in an implementation-defined manner.

ANS: (d)

**9.8 Printing with Field Widths and Precision**

9.32 What does the 4 signify in the follwing statement?

printf( "%4i\n", 123 );

(a) degree of exponentiation

(b) floating point precision

(c) a field width

(d) none of these

ANS: (c)

9.33 What would be the output of the following statement?

printf( "%4d", 123456 );

(a) 1234

(b) 2345

(c) 3456

(d) none of these

ANS: (d)

9.34 The statement

printf( "%\*.\*f", 7, 2, 98.736 );

uses \_\_\_\_\_\_\_ for the precision, \_\_\_\_\_\_\_\_\_\_ for the field width and outputs the value 98.74 \_\_\_\_\_\_\_\_\_\_.

a) 7, 2, left justified

b) 2, 7, left justified

c) 2, 7, right justified

d) 7, 2, right justified

ANS: (c)

9.35 If the field width is larger than the data being printed, the data will normally be \_\_\_\_\_\_\_\_\_\_ within that field.

a) truncated

b) right justified

c) centered

d) left justified

ANS: (b)

**9.9 Using Flags in the printf Format-Control String**

9.36 Which of the following is *not* a flag that can be used in the printf format-control string?

(a) 0

(b) +

(c) /

(d) #

ANS: (c)

9.37 Which of the following is the correct way to output the value of 4 left justified?

(a) printf( "%i", 4 );

(b) printf( "%-i", 4 );

(c) printf( "4%i", 4 );

(d) printf( "4-%i", 4 );

ANS: (b)

9.38 Which is *not* a format control string flag?

a) –

b) space

c) newline

d) #

ANS: (c)

9.39 Which statement is *false* with regard to format control string flags?

a) The plus sign is only displayed when the + flag is used.

b) The minus sign is always displayed (when the value being printed is negative).

c) To use a flag in a format control string, place the flag immediately to the left of the percent sign.

d) Use the 0 flag to pad a field with leading zeros.

ANS: (c)

**9.10 Printing Literals and Escape Sequences**

9.40 Which of the following is *not* an escape sequence?

(a) \a

(b) \b

(c) \c

(d) none of these

ANS: (c)

9.41 What does the \\ do when used in a format control string?

(a) It outputs the backslash character

(b) It delimits comments

(c) both a and b

(d) none of these

ANS: (a)

9.42 Which statement about an escape sequence is *false*?

a) \r moves the cursor to the beginning of the next line.

b) \b moves the cursor back one position on the current line.

c) \f moves the cursor to the start of the next logical page.

d) \v moves the cursor to the next vertical tab position.

ANS: (a)

**9.11 Formatting Input with scanf**

9.43 A field width \_\_\_\_\_\_\_\_ be included in the format control string of the scanf statement.

(a) can

(b) cannot

(c) must

(d) none of these

ANS: (a)

9.44 The \_\_\_\_\_\_\_\_ character reads data from the input stream and discards the data.

(a) *%*

(b) conversion specifier

(c) assignment suppression

(d) //

ANS: (c)

9.45 Which is *not* an input formatting capability of scanf?

a) inputting all types of data

b) inputting specific characters from an input stream

c) skipping specific characters in the input stream

d) replacing specific characters in the input stream

ANS: (d)

9.46 The %i scanf conversion specifier is *not* capable of inputting which type of data?

a) hexadecimal

b) binary

c) decimal

d) octal

ANS: (b)

9.47 Which statement about scan sets is *false*?

a) A scan set is a set of characters enclosed in parentheses and preceded by a percent sign in the format control string.

b) A scan set scans the characters in the input stream looking only for those characters that match the characters contained in the scan set.

c) Each time a character in the input stream matches a character in the scan set, the input stream character is stored in the scan set’s corresponding argument—a pointer to a character array.

d) The scan set stops inputting characters when a character that is not contained in the scan set is encountered.

ANS: (a)

**9.12 Secure C Programming**

9.48 Which of the following statements is *false*?

(a) In general, as you study any programming language, if the language specification says that doing something can lead to undefined behavior, avoid doing it to prevent security vulnerabilities.

(b) The C standard specifies the exact behavior for every case in which incorrect library-function arguments are passed to a library function.

(c) Undefined behaviors can occur when using printf with improperly formed conversion specifications.

(d)None of the above.

ANS: (b)