

1. What is assigned to the variable `a` given the statement below with the following assumptions: `x = 10`, `y = 7`, and `z`, `a`, and `b` are all `int` variables.

```
a = x >= y;
```

- a. 10
 - b. 7
 - c. The string "`x >= y`"
 - d. 1
2. When a relational expression is false, it has the value ____.
- a. one
 - b. zero
 - c. zero, one, or minus one
 - d. less than zero
3. What will following segment of code output?

```
int x = 5;
if (x = 2)
    cout << "This is true!" << endl;
else
    cout << "This is false!" << endl;
    cout << "This is all folks!" << endl;
```

- a. This is true!
 - b. This is false!
 - c. This is true!
This is false!
 - d. This is true!
This is all folks!
4. What will the following segment of code output? You can assume the user enters a grade of 90 from the keyboard.

```
cout << "Enter a test score: ";
cin >> test_score;
if (test_score < 60);
    cout << "You failed the test!" << endl;
if (test_score > 60)
    cout << "You passed the test!" << endl;
else
    cout << "You need to study for the next test!";
```

- a. You failed the test!
 - b. You passed the test!
 - c. You failed the test!
You passed the test!
 - d. You failed the test!
You did poorly on the test!
5. This operator represents the logical AND.
- a. ++
 - b. ||
 - c. &&
 - d. @

6. This operator takes an operand and reverses its truth or falsehood.
- a. ||
 - b. relational
 - c. arithmetic
 - d. !
 - e. None of these
7. What is the output of the following segment of code if 4 is input by the user when asked to enter a number?

```
int num;
int total = 0;
cout << "Enter a number from 1 to 10: ";
cin >> num;
switch (num)
{
    case 1:
    case 2:    total = 5;
    case 3:    total = 10;
    case 4:    total = total + 3;
    case 8:    total = total + 6;
    default:   total = total + 4;
}
cout << total << endl;
```

- a. 0
 - b. 3
 - c. 13
 - d. 28
 - e. None of these
8. What will the following segment of code output?

```
score = 40;
if (score > 95)
    cout << "Congratulations!\n";
    cout << "That's a high score!\n";
    cout << "This is a test question!" << endl;
```

- a. This is a test question!
 - b. That's a high score!
This is a test question!
 - c. Congratulations!
That's a high score!
This is a test question!
 - d. Congratulations!
That's a high score!
 - e. None of these
9. This operator is used in C++ to represent equality.
- a. =
 - b. ><
 - c. !!
 - d. ==
 - e. None of these

10. Assuming x is 5, y is 6, and z is 8, which of the following is false?

1. `x == 5;`
2. `7 <= (x + 2);`
3. `z <= 4;`
4. `(1 + x) != y;`
5. `z >= 8;`
6. `x >= 0;`
7. `x <= (y * 2)`

- a. 3, 4, 6, 7 are False
- b. Only 5 is False
- c. 3 and 4 are False
- d. All are False
- e. None of these

11. If you intend to place a block of statements within an if statement, you must place these around the block.

- a. parentheses ()
- b. square brackets []
- c. quotation marks ? ?
- d. curly braces { }
- e. None of these

12. What will the following segment of code output if 11 is entered at the keyboard?

```
int number;
cin >> number;
if (number > 0)
    cout << "C++";
else
    cout << "Soccer";
    cout << " is ";
    cout << "fun" << endl;
```

- a. C++ is fun
- b. Soccer is fun
- c. C++
- d. C++fun
- e. Soccerfun

13. Given that x = 2, y = 1, and z = 0, what will the following cout statement display?

```
cout << "answer = " << (x || !y && z) << endl;
```

- a. answer = 0
- b. answer = 1
- c. answer = 2
- d. None of these

14. What will the following program segment display?

```
int funny = 7, serious = 15;
funny = serious % 2;
if (funny != 1)
{
    funny = 0;
    serious = 0;
}
else if (funny == 2)
{
    funny = 10;
    serious = 10;
}
else
{
    funny = 1;
    serious = 1;
}
cout << funny << "   " << serious << endl;
```

- a. 7 15
- b. 0 0
- c. 10 10
- d. 1 1

15. What will the following program display?

```
#include <iostream>
using namespace std;
int main()
{
    int a = 0, b = 2, x = 4, y = 0;
    cout << (a == b) << " ";
    cout << (a != b) << " ";
    cout << (b <= x) << " ";
    cout << (y > a) << endl;
    return 0;
}
```

- a. 0 1 1 0
- b. 0 0 1 0
- c. 1 1 0 1
- d. 1 0 0 1
- e. None of these

16. What is the output of the following code segment?

```
n = 1;
while (n <= 5)
    cout << n << ' ';
    n++;
```

- a. 1 2 3 4 5
- b. 1 1 1... and on forever
- c. 2 3 4 5 6
- d. 1 2 3 4
- e. 2 3 4 5

17. Given the following code segment, what is output after "result = "?

```
int x = 1, y = 1, z = 1;
y = y + z;
x = x + y;
cout << "result = "
      << (x < y ? y : x)
      << endl;
```

- a. 0
- b. 1
- c. 2
- d. 3
- e. None of these

18. What is the output of the following code?

```
int w = 98;
int x = 99;
int y = 0;
int z = 1;
if (x >= 99)
{
    if (x < 99)
        cout << y << endl;
    else
        cout << z << endl;
}
else
{
    if (x == 99)
        cout << x << endl;
    else
        cout << w << endl;
}
```

- a. 98
- b. 99
- c. 0
- d. 1

19. Which value can be entered to cause the following code segment to display the message "That number is acceptable."

```
int number;
cin >> number;
if (number > 10 && number < 100)
    cout << "That number is acceptable.\n";
else
    cout << "That number is not acceptable.\n";
```

- a. 100
- b. 10
- c. 99
- d. 0
- e. All of these

20. Which line in the following program will cause a compiler error?

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int number = 5;
7
8      if (number >= 0 && <= 100)
9          cout << "passed.\n";
10     else
11         cout << "failed.\n";
12     return 0;
13 }
```

- a. 6
- b. 8
- c. 10
- d. 9

21. What will be the value of result after the following code has been executed?

```
int a = 60;
int b = 15;
int result = 10;
if (a = b)
    result *= 2;
```

- a. 10
- b. 120
- c. 20
- d. This code will not compile

22. What is the output of the following code segment?

```
n = 1;
for ( ; n <= 5; )
    cout << n << '  ';
    n++;
```

- a. 1 2 3 4 5
- b. 1 1 1 ... and on forever
- c. 2 3 4 5 6
- d. 1 2 3 4
- e. 2 3 4 5

23. The do-while loop is considered a(n) _____ loop.

- a. pre-test
- b. post-test
- c. infinite
- d. limited
- e. None of these

24. This statement causes a loop to terminate early.

- a. stop
- b. break
- c. null
- d. terminate

25. What will the following loop display?

```
int x = 0;
while (x < 5)
{
    cout << x << endl;
    x++;
}
```

- a. 0
1
2
3
4
5
- b. 0
1
2
3
4
- c. 0 1 2 3 4
- d. The loop will display numbers starting at 0, for infinity.

26. What will the following code display?

```
int number = 6;
number++;
cout << number << endl;
```

- a. 6
- b. 5
- c. 7
- d. 0

27. To allow file access in a program, you must #include this header file.

- a. file
- b. fileaccess
- c. fstream
- d. cfile

28. What will the following code display?

```
int x = 0;
for (int count = 0; count < 3; count++)
    x += count;
cout << x << endl;
```

- a. 0
1
2
- b. 0
- c. 6
- d. 3

29. To write data to a file, you define an object of this data type.

- a. outputFile
- b. ifstream
- c. fstream
- d. ofstream

30. To read data from a file, you define an object of this data type.
- a. inputFile
 - b. ifstream
 - c. fstream
 - d. ofstream

31. How many times will the following loop display "Hello"?

```
for (int i = 0; i < 20; i++)  
    cout << "Hello!" << endl;
```

- a. 20
- b. 19
- c. 21
- d. An infinite number of times

32. How many times will the following loop display "Hello"?

```
for (int i = 1; i < 20; i++)  
    cout << "Hello!" << endl;
```

- a. 20
- b. 19
- c. 21
- d. An infinite number of times

33. This is a collection of statements that performs a specific task.

- a. infinite loop
- b. variable
- c. constant
- d. function
- e. None of these

34. A function _____ contains the statements that make up the function.

- a. definition
- b. prototype
- c. call
- d. expression
- e. parameter list

35. A function can have zero to many parameters, and it can return this many values.

- a. zero to many
- b. no
- c. only one
- d. a maximum of ten
- e. None of these

36. A function is executed when it is

- a. defined
- b. prototyped
- c. declared
- d. called
- e. None of these

37. When used as parameters, these types of variables allow a function to access the parameter's original argument.
- a. reference
 - b. floating-point
 - c. counter
 - d. undeclared
 - e. None of these
38. This statement causes a function to end.
- a. end
 - b. terminate
 - c. return
 - d. release
 - e. None of these
39. _____ functions may have the same name, as long as their parameter lists are different.
- a. Only two
 - b. Two or more
 - c. Zero
 - d. Un-prototyped
 - e. None of these
40. This function causes a program to terminate, regardless of which function or control mechanism is executing.
- a. terminate()
 - b. return()
 - c. continue()
 - d. exit()
 - e. None of these
41. Given the following function definition

```
void calc (int a, int& b)
{
    int c;

    c = a + 2;
    a = a * 3;
    b = c + a;
}
```

What is the output of the following code fragment that invokes calc?
(All variables are of type *int*)

```
x = 1;
y = 2;
z = 3;
calc(x, y);
cout << x << " " << y << " " << z << endl;
```

- a. 1 2 3
- b. 1 6 3
- c. 3 6 3
- d. 1 14 9
- e. None of these

42. What is the output of the following program?

```
#include <iostream>
using namespace std;

void showDub(int);

int main()
{
    int x = 2;
    showDub(x);
    cout << x << endl;
    return 0;
}

void showDub(int num)
{
    cout << (num * 2) << endl;
}
```

- | | |
|------|------|
| a. 2 | c. 2 |
| 2 | 4 |
| b. 4 | d. 4 |
| 2 | 4 |

43. What is the output of the following program?

```
#include <iostream>
using namespace std;

void doSomething(int);

int main()
{
    int x = 2;

    cout << x << endl;
    doSomething(x);
    cout << x << endl;
    return 0;
}

void doSomething(int num)
{
    num = 0;
    cout << num << endl;
}
```

- | | |
|------|------|
| a. 2 | c. 0 |
| 0 | 0 |
| 2 | 0 |
| b. 2 | d. 2 |
| 2 | 0 |
| 2 | 0 |

44. What is the output of the following program?

```
#include <iostream>
using namespace std;

void doSomething(int&);

int main()
{
    int x = 2;

    cout << x << endl;
    doSomething(x);
    cout << x << endl;
    return 0;
}

void doSomething(int& num)
{
    num = 0;
    cout << num << endl;
}
```

- | | |
|------|------|
| a. 2 | c. 0 |
| 0 | 0 |
| 2 | 0 |
| b. 2 | d. 2 |
| 2 | 0 |
| 2 | 0 |

45. Which line in the following program contains the prototype for the showDub function?

```
1 #include <iostream>
2 using namespace std;
3
4 void showDub(int);
5
6 int main()
7 {
8     int x = 2;
9
10    showDub(x);
11    cout << x << endl;
12    return 0;
13 }
14
15 void showDub(int num)
16 {
17     cout << (num * 2) << endl;
18 }
```

- | | |
|------|-------|
| a. 4 | c. 10 |
| b. 6 | d. 15 |

46. If a function is called more than once in a program, the values stored in the function's local variables do not _____ between function calls.
- a. persist
 - b. execute
 - c. communicate
 - d. Change

47. Which line in the following program contains the header for the showDub function?

```
1  #include <iostream>
2  using namespace std;
3
4  void showDub(int);
5
6  int main()
7  {
8      int x = 2;
9
10     showDub(x);
11     cout << x << endl;
12     return 0;
13 }
14
15 void showDub(int num)
16 {
17     cout << (num * 2) << endl;
18 }
```

- a. 4
- b. 6
- c. 10
- d. 15

48. Which line in the following program contains a call to the showDub function?

```
1  #include <iostream>
2  using namespace std;
3
4  void showDub(int);
5
6  int main()
7  {
8      int x = 2;
9
10     showDub(x);
11     cout << x << endl;
12     return 0;
13 }
14
15 void showDub(int num)
16 {
17     cout << (num * 2) << endl;
18 }
```

- a. 4
- b. 6
- c. 10
- d. 15

49. What is the output of the following program?

```
#include <iostream>
using namespace std;

int getValue(int);

int main()
{
    int x = 2;

    cout << getValue(x) << endl;
    return 0;
}

int getValue(int num)
{
    return num + 5;
}
```

- a. 5
- b. 2
- c. 7
- d. "getValue(x)"

50. Unlike regular variables, these can hold multiple values.

- a. constants
- b. named constants
- c. arrays
- d. floating-point variables
- e. None of these

51. The individual values contained in array are known as

- a. parts
- b. elements
- c. numbers
- d. constants
- e. None of these

52. To access an array element, use the array name and the element's

- a. data type
- b. subscript
- c. name
- d. value
- e. None of these

53. The statement: `int grades[] = { 100, 90, 99, 80};` shows an example of

- a. default arguments
- b. an illegal array declaration
- c. an illegal array initialization
- d. implicit array sizing
- e. None of these

54. A two-dimensional array can have elements of _____ data type(s).

- a. one
- b. two
- c. four
- d. Any of these
- e. None of these

55. How many elements does the following array have?

```
int bugs[1000];
```

- a. 1000
- b. 999
- c. 1001
- d. Cannot tell from the code

56. What will the following code display?

```
int numbers[] = {99, 87, 66, 55, 101 };  
cout << numbers[3] << endl;
```

- a. 55
- b. 66
- c. 101
- d. 87

57. Which statement correctly defines a vector object for holding integers?

- a. `vector v<int>;`
- b. `int vector v;`
- c. `int<vector> v;`
- d. `vector<int> v;`

58. What will the following code display?

```
int numbers[] = { 99, 87, 66, 55, 101 };  
for (int i = 1; i < 4; i++)  
    cout << numbers[i] << endl;
```

- a. 99
87
66
55
101
- b. 87
66
55
101
- c. 87
66
55
- d. Nothing. This code has an error.

59. What will the following code do?

```
const int SIZE = 5;  
double x[SIZE];  
for(int i = 2; i <= SIZE; i++)  
{  
    x[i] = 0.0;  
}
```

- a. Each element in the array is initialized to 0.0
- b. Each element in the array, except the first, is initialized to 0.0
- c. Each element in the array, except the first and the last, is initialized to 0.0
- d. An error will occur when the code runs

60. What does the following statement do?

```
vector<int> v(10);
```

- a. It creates a vector object and initializes all of its elements to the value 10.
- b. It creates a vector object with a starting size of 10.
- c. It creates a vector object and initializes the first element with the value 10.
- d. It creates a vector object that can store only values of 10 or less.

61. This vector function is used to insert an item into a vector.

- a. `insert_item`
- b. `add_item`
- c. `store`
- d. `push_back`

62. What does the following statement do?

```
vector<int> v(10, 2);
```

- a. It creates a vector object and initializes all the first two elements with the values 10 and 2.
- b. It creates a vector object with a starting size of 2 and the first element initialized with the value 10.
- c. It creates a vector object with a starting size of 10 and the first element initialized with the value 2.
- d. It creates a vector object with a starting size of 10 and all elements are initialized with the value 2.

63. What will the following code display?

```
int numbers[4] = { 99, 87 };  
cout << numbers[3] << endl;
```

- a. 87
- b. 0
- c. garbage
- d. This code will not compile

64. This vector function returns the number of elements in a vector.

- a. `size`
- b. `num_elements`
- c. `elements`
- d. `length`

65. This vector function removes an item from a vector.

- a. `remove_item`
- b. `delete_item`
- c. `erase`
- d. `pop_back`

66. This vector function returns true if the vector has no elements.

- a. `has_no_elements`
- b. `null_size`
- c. `empty`
- d. `is_empty`

67. The _____ , also known as the address operator, returns the memory address of a variable.
- asterisk (*)
 - ampersand (&)
 - percent sign (%)
 - exclamation point (!)
 - None of these
68. With pointer variables, you can _____ manipulate data stored in other variables.
- never
 - seldom
 - indirectly
 - All of these
 - None of these
69. The statement
- ```
int *ptr;
```
- has the same meaning as
- int ptr;
  - \*int ptr;
  - int ptr\*;
  - int\* ptr;
  - None of these
70. When you work with a dereferenced pointer, you are actually working with:
- a variable whose memory has been deallocated
  - a copy of the value pointed to by the pointer variable
  - the actual value of the variable whose address is stored in the pointer variable
  - All of these
  - None of these
71. These can be used as pointers.
- Array names
  - Numeric constants
  - Punctuation marks
  - All of these
  - None of these
72. The contents of pointer variables may be changed with mathematical statements that perform:
- all mathematical operations that are legal in C++
  - multiplication and division
  - addition and subtraction
  - b and c
73. A pointer may be initialized with
- the address of an existing object
  - the value of an integer variable
  - the value of a floating point variable
  - all of these
  - None of these



74. What does the following statement do?

```
double *num2;
```

- a. Declares a double variable named num2.
- b. Declares and initializes an pointer variable named num2.
- c. Initializes a variable named \*num2.
- d. Declares a pointer variable named num2.
- e. None of these

75. When the less than ( < ) operator is used between two pointer variables, the expression is testing whether

- a. the value pointed to by the first is less than the value pointed to by the second
- b. the value pointed to by the first is greater than the value pointed to by the second
- c. the address of the first variable comes before the address of the second variable in the computer's memory
- d. the first variable was declared before the second variable
- e. None of these

76. Look at the following statement:

```
sum += *array++;
```

This statement...

- a. is illegal in C++
- b. will always result in a compiler error
- c. assigns the dereferenced pointer's value, then increments the pointer's address
- d. increments the dereferenced pointer's value by one, then assigns that value
- e. None of these

77. Use the delete operator only on pointers that were

- a. never used
- b. not correctly initialized
- c. created with the new operator
- d. dereferenced inappropriately
- e. None of these

78. A function may return a pointer, but the programmer must ensure that the pointer

- a. still points to a valid object after the function ends
- b. has not been assigned an address
- c. was received as a parameter by the function
- d. has not previously been returned by another function
- e. None of these

79. Which of the following statements is not valid C++ code?

- a. `int ptr = &num1;`
- b. `int ptr = int *num1;`
- c. `float num1 = &ptr2;`
- d. All of these are valid
- e. All of these are invalid

80. Which of the following statements deletes memory that has been dynamically allocated for an array?
- a. `int array = delete memory;`
  - b. `int delete[ ];`
  - c. `delete [] array;`
  - d. `new array = delete;`
  - e. None of these

81. When this is placed in front of a variable name, it returns the address of that variable.
- a. asterisk ( \* )
  - b. conditional operator
  - c. ampersand ( & )
  - d. semicolon ( ; )
  - e. None of these

82. What will the following statement output?

```
cout << &num1;
```

- a. The value stored in the variable called num1.
- b. The memory address of the variable called num1.
- c. The number 1.
- d. The string "&num1".
- e. None of these

83. A pointer variable is designed to store

- a. any legal C++ value.
- b. only floating-point values.
- c. a memory address.
- d. an integer.
- e. None of these

84. Look at the following statement.

```
int *ptr;
```

In this statement, what does the word `int` mean?

- a. the variable named `*ptr` will store an integer value
  - b. the variable named `*ptr` will store an asterisk and an integer value
  - c. `ptr` is a pointer variable that will store the address of an integer variable
  - d. All of these
  - e. None of these
85. Assuming `ptr` is a pointer variable, what will the following statement output?

```
cout << *ptr;
```

- a. the value stored in the variable whose address is contained in `ptr`.
- b. the string `"*ptr"`.
- c. the address of the variable stored in `ptr`.
- d. the address of the variable whose address is stored in `ptr`.
- e. None of these.

86. The \_\_\_\_\_ and \_\_\_\_\_ operators can be used to increment or decrement a pointer variable.
- addition, subtraction
  - modulus, division
  - ++, --
  - All of these
  - None of these
87. Not all arithmetic operations may be performed on pointers. For example, you cannot \_\_\_\_\_ or \_\_\_\_\_ a pointer.
- multiply, divide
  - add, subtract
  - +=, -=
  - increment, decrement
  - None of these
88. Which statement displays the address of the variable num1?
- cout << num1;
  - cout << \*num1;
  - cin >> &num1;
  - cout << &num1;
  - None of these
89. The statement cin >> \*num3;
- stores the keyboard input into the variable num3.
  - stores the keyboard input into the pointer called num3.
  - is illegal in C++.
  - stores the keyboard input into the variable pointed to by num3.
  - None of these.
90. Dynamic memory allocation occurs
- when a new variable is created by the compiler
  - when a new variable is created at runtime
  - when a pointer fails to dereference the right variable
  - when a pointer is assigned an incorrect address
  - None of these
91. The statement int \*ptr = new int;
- results in a compiler error.
  - assigns an integer less than 32767 to the variable named ptr.
  - assigns an address to the variable named ptr.
  - creates a new pointer named int.
  - None of these
92. When using the new operator with an older compiler, it is good practice to:
- test the pointer for the NULL address
  - use a preprocessor directive
  - clear the data from the old operator
  - All of these
93. Every byte in the computer's memory is assigned a unique
- pointer
  - address
  - dynamic allocation
  - name

94. When you pass a pointer as an argument to a function, you must
- redeclare the pointer variable in the function call
  - dereference the pointer variable in the function prototype
  - use the `#include<func_ptr.h>` statement
  - None of these
95. A pointer variable may be initialized with
- any non-zero integer value.
  - any address in the computer's memory.
  - an address less than 0
  - a and c only.
96. If a variable uses more than one byte of memory, for pointer purposes its address is:
- the address of the last byte of storage.
  - the average of the addresses used to store the variable.
  - the address of the first byte of storage.
  - general delivery.

97. What will the following code output?
- ```
int number = 22;
int *var = &number;
cout << *var << endl;
```
- The address of the number variable
 - 22
 - An asterisk followed by 22
 - An asterisk followed by the address of the number variable

98. What will the following code output?
- ```
int number = 22;
int *var = &number;
cout << var << endl;
```
- The address of the number variable
  - 22
  - An asterisk followed by 22
  - An asterisk followed by the address of the number variable

99. What will the following code output:
- ```
int *numbers = new int[5];
for (int i = 0; i <= 4; i++)
    *(numbers + i) = i;
cout << numbers[2] << endl;
```
- 0
 - 3
 - 1
 - 2

100. Look at the following code:
- ```
int numbers[] = {0, 1, 2, 3, 4 };
int *ptr = numbers;
ptr++;
```

After this code executes, which of the following statements is true?

- ptr will hold the address of numbers[0]
- ptr will hold the address of the 2nd byte within the element numbers[0]
- ptr will hold the address of numbers[1]
- This code will not compile.