Pointers

MULTIPLE CHOICE

1. The _____ B _____, also known as the address operator, returns the memory address of a variable.
   a. asterisk (*)
   b. ampersand (&)
   c. percent sign (%)
   d. exclamation point (!)
   e. None of these

2. With pointer variables, you can _____ C _____ manipulate data stored in other variables.
   a. never
   b. seldom
   c. indirectly
   d. All of these
   e. None of these

3. The statement

   ```
   int *ptr;
   ```

   has the same meaning as

   a. `int ptr;`
   b. `*int ptr;`
   c. `int ptr*;`
   d. `int* ptr;`
   e. None of these

4. When you work with a dereferenced pointer, you are actually working with:
   a. a variable whose memory has been deallocated
   b. a copy of the value pointed to by the pointer variable
   c. the actual value of the variable whose address is stored in the pointer variable
   d. All of these
   e. None of these

5. These can be used as pointers.
   a. Array names
   b. Numeric constants
   c. Punctuation marks
   d. All of these
   e. None of these

6. The contents of pointer variables may be changed with mathematical statements that perform:
   a. all mathematical operations that are legal in C++
   b. multiplication and division
   c. addition and subtraction
   d. b and c

e. None of these

7. A pointer may be initialized with
   a. the address of an existing object
   b. the value of an integer variable
   c. the value of a floating point variable
   d. all of these
   e. None of these

8. 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

9. 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

10. 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

11. 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

12. 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

13. 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

14. 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

15. 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

16. 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

17. 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

18. 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

19. 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.

20. The _____ and _____ operators can be used to increment or decrement a pointer variable.
a. addition, subtraction
b. modulus, division
c. ++, --
d. All of these
e. None of these

21. Not all arithmetic operations may be performed on pointers. For example, you cannot _____ or _____ a pointer.
a. multiply, divide
b. add, subtract
c. +=, -=
d. increment, decrement
e. None of these

22. Which statement displays the address of the variable `num1`?
a. `cout << num1;`
b. `cout << *num1;`
c. `cin >> &num1;`
d. `cout << &num1;`
e. None of these

23. The statement `cin >> *num3;`
a. stores the keyboard input into the variable `num3`.
b. stores the keyboard input into the pointer called `num3`.
c. is illegal in C++.
d. stores the keyboard input into the variable pointed to by `num3`.
e. None of these.

24. Dynamic memory allocation occurs
a. when a new variable is created by the compiler
b. when a new variable is created at runtime
c. when a pointer fails to dereference the right variable
d. when a pointer is assigned an incorrect address

e. None of these

25. The statement `int *ptr = new int;`
    a. results in a compiler error.
    b. assigns an integer less than 32767 to the variable named `ptr`.
    c. assigns an address to the variable named `ptr`.
    d. creates a new pointer named `int`.
    e. None of these

26. When using the `new` operator with an older compiler, it is good practice to:
    a. test the pointer for the NULL address
    b. use a preprocessor directive
    c. clear the data from the `old` operator
    d. All of these
    e. None of these

27. Every byte in the computer's memory is assigned a unique
    a. pointer
    b. address
    c. dynamic allocation
    d. name
    e. None of these

28. When you pass a pointer as an argument to a function, you must
    a. redeclare the pointer variable in the function call
    b. dereference the pointer variable in the function prototype
    c. use the `#include<func_ptr.h>` statement
    d. not dereference the pointer in the function's body
    e. None of these

29. A pointer variable may be initialized with
    a. any non-zero integer value.
    b. any address in the computer's memory.
    c. an address less than 0
    d. a and c only.
    e. None of these.

30. If a variable uses more than one byte of memory, for pointer purposes its address is:
    a. the address of the last byte of storage.
    b. the average of the addresses used to store the variable.
    c. the address of the first byte of storage.
    d. general delivery.
    e. None of these.

31. What will the following code output?

```
int number = 22;
int *var = &number;
cout << *var << endl;
```

a. The address of the `number` variable
b. 22
c. An asterisk followed by 22
d. An asterisk followed by the address of the `number` variable

32. What will the following code output?

```
int number = 22;
int *var = &number;
cout << var << endl;
```

a. The address of the `number` variable
b. 22
c. An asterisk followed by 22
d. An asterisk followed by the address of the `number` variable

33. 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;
```

a. Five memory addresses
b. 0
c. 3
d. 2
e. 1

34. 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?

a. `ptr` will hold the address of `numbers[0]`
b. `ptr` will hold the address of the 2nd byte within the element `numbers[0]`
c. `ptr` will hold the address of `numbers[1]`
d. This code will not compile.