Sample Exam Week 02

CSE 232 (Introduction to Programming II)

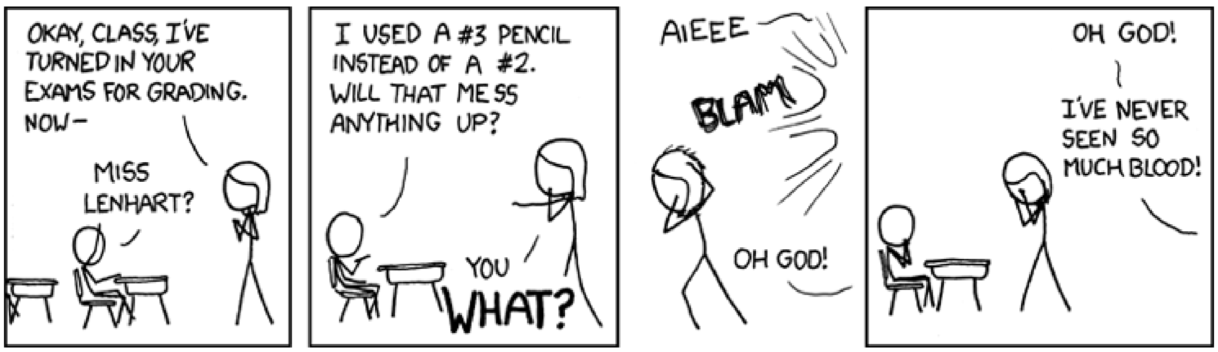
VERSION A

Full Name: ....

Student Number: ....

Instructions:

- DO NOT START/OPEN THE EXAM UNTIL TOLD TO DO SO.
- You may however write and bubble in your name, student number and exam **VERSION/FORM NUMBER** (with a #2 pencil) on the front of the printed exam and bubble sheet prior to the exam start. This exam is Version A. Your section doesn't matter and can be ignored.
- Present your MSU ID (or other photo ID) when returning your bubble sheet and printed exam.
- Only choose one option for each question. Please mark the chosen option in both this printed exam and the bubble sheet.
- Assume any needed `#include`s and `using std::...;` namespace declarations are performed for the code samples.
- Every question is worth the same amount of points. There are 55 questions, but you only need 50 questions correct for a perfect score.
- No electronics are allowed to be used or worn during the exam. This means smart-watches, phones and headphones need to be placed away in your bag.
- The exam is open note, meaning that any paper material (notes, slides, prior exams, assignments, books, etc.) are all allowed. Please place all such material on your desk prior to the start of the exam, (so you won't need to rummage in your bag during the exam).
- If you have any questions during the exam or finish the exam early, please raise your hand and a proctor will attend you.


OKAY, CLASS, I'VE TURNED IN YOUR EXAMS FOR GRADING. NOW—
MISS LENHART?
I USED A #3 PENCIL INSTEAD OF A #2. WILL THAT MESS ANYTHING UP?
YOU WHAT?
AIEEE
BLAM
OH GOD!
OH GOD!
I'VE NEVER SEEN SO MUCH BLOOD!

http://xkcd.com/499/

1. Which of the following are safe to do with a null pointer?

   (a) Make a copy of it.
   (b) Store a memory address in it.
   (c) Make a const reference to it.
   (d) Dereference it.
   (e) Only (a) and (b) are safe.
   (f) Only (a), (b) and (c) are safe.
   (g) All of (a), (b), (c), and (d) are safe.

2. What benefit is there to using a pointer instead of a reference?

   (a) A pointer can more safely refer to an object because it can be declared const.
   (b) A pointer can point to objects and fundamental/primitive types, whereas a reference can only refer to objects of custom classes.
   (c) A pointer can be used to do bit-wise arithmetic, where as a reference is limited to integer arithmetic.
   (d) A pointer can refer to two objects at the same time because it can hold multiple addresses, whereas a reference can only return to one object.
   (e) None of the above.

3. How do you stop fall-through behavior?

   (a) By ensuring that you have a default case.
   (b) You cant stop fall-through, it is mandated by the C++ language.
   (c) By ensuring that you have a break at the end of each case.
   (d) By using a conditional statement to check for const expressions.
   (e) By specifying a zero case for your conditional expression.
   (f) None of the above

4. What does the `const` in the following declaration imply about `x`?
   `long * const x = &y;`

   (a) That the value of x (the address) cannot be changed to point at a different position in memory.
   (b) That the value pointed at by x (the long) cannot be changed to be a different value.
   (c) Both of the above.
   (d) Neither of the above.

5. What is the difference between `x` and `y`?
   `for (auto x : vec) ...`
   `for (auto & y : vec) ...`

   (a) `x` is a copy of each element, where as `y` is a reference.
   (b) If `y` is altered, the element in `vec` changes. Not so for `x`.
   (c) Both of the above are true.
   (d) None of the above are true.

6. When should you use a pointer instead of a reference.

   (a) When you need to perform pointer arithmetic.
   (b) When a library function you need requires a pointer argument.
   (c) When you need to store the address of an object.
   (d) All of the above are true.
   (e) None of the above are true.

7. Which of the following permit 0 or more statements?

   (a) The contents of a block statement
   (b) The body of an if statement
   (c) A function's body
   (d) Pre-processor statements
   (e) (b) and (c) are both correct.
   (f) None of the above are correct.

8. Why do we recommend against using the `long` and `long long` types for large integers?

   (a) Because using them is undefined behavior
   (b) Because `int32_t` and `int64_t` provide more guarantees about their capacity
   (c) Because they are unsigned and hence can't represent negative integers
   (d) Because they take up more characters in your program
   (e) Because they take up too much memory and slow programs down

9. Which of the following statements would cause a compilation error if included after the following code?

```
int x = 4; int y = 7;
int const *ptr = &x;
```

   (a) `ptr = &y;`
   (b) `++x;`
   (c) `++ptr;`
   (d) Both (a) and (b) will cause compilation errors
   (e) Both (a) and (c) will cause compilation errors
   (f) Both (b) and (c) will cause compilation errors
   (g) All of (a), (b), and (c) will cause compilation errors
   (h) None of the above will cause an error

10. Why can you not declare a reference?

   (a) Because a reference must be able to change the value it refers to
   (b) Because a reference can only be made of a const value
   (c) You can declare a reference, but only if the reference is in a const member function
   (d) Because a reference can't exist without referring to something
   (e) None of the above

11. Can you take the address of a pointer?

   (a) No, but you can make a reference to them.
   (b) No, pointers don't exist apart from the objects they point at
   (c) Yes, pointers like all objects, have addresses
   (d) Yes, but only pointers that aren't null

12. What will be the output of the following code?

```
int main() {
   int a[3] = {1, 2, 3};
   int *p = a;
   cout << p << '-' << a;
}
```

   (a) undefined
   (b) Two different addresses are printed
   (c) Run time error
   (d) Same address is printed twice
   (e) Compile time error

13. What will be the output of the following code?

```
int main() {
  int a[3] = {1, 2, 3};
  int *p = a;
  cout << *(a+1) << '-' << p[1];
}
```

   (a) Different addresses are printed
   (b) 2-2
   (c) Compile time error
   (d) Run time error
   (e) 1-1

14. What is the difference between
`for (auto x : xs) ...`
and
`for (auto const & x : xs) ...`

  (a) Only the second range-based for loop can work if `xs` is const.
  (b) Only the first loop will compile if `x` in a fundamental type.
  (c) The second range-based for loop is able to change `xs`.
  (d) The first range-based for loop copies each element.
  (e) They have the same behavior if if `xs` is a vector of ints, but not a vector of strings.
  (f) All of the above.

15. What does the following code do?
`int a = 12; auto b = & (&a);`

  (a) It takes the address of a reference to an int.
  (b) It makes an object of type auto.
  (c) It makes a copy of the value 12.
  (d) It generates a function that returns an int.
  (e) It makes a reference to an address of an int.
  (f) None of the above.

16. Which of the following C++ keywords causes a loop to immediately repeat (skipping the rest of the loop's body)?

  (a) `continue`
  (b) `break`
  (c) `repeat`
  (d) `catch`
  (e) `switch`
  (f) `goto`

17. What character is used to denote the end of a statement?

  (a) >
  (b) ′
  (c)]
  (d) ;
  (e))
  (f) ”
  (g) None of the previous.

18. What does the following code output?
```
int x = 9;
while (x = 4) {
  cout << x;
  cout << x++;
  if (7) break;
  cout << x;
}
cout << x;
```

  (a) `9`
  (b) `4`
  (c) It never ends.
  (d) `9101010`
  (e) `991010`
  (f) It doesn't compile.
  (g) `455`
  (h) `445`

19. A for loop has 4 parts (`for (a; b; c) body`). After which circumstances does the statement at position `c` run?

  (a) After a continue statement executes.
  (b) After a break statement executes.
  (c) After the body finishes normally.
  (d) Only (a) and (b) are correct.
  (e) Only (a) and (c) are correct.
  (f) All of the above are correct.

20. Which of the following (5) lines would generate a syntax error if included in a C++ file?

  (a) `while (cin) {`
  (b) `for (;;) {`
  (c) `(a = b) = 3;`
  (d) `while (true) ;`
  (e) `x = (y >= z);`
  (f) All of the above lines do not generate syntax errors.

21. Which of the following expressions do **NOT** evaluate to a true value?
    (a) `true`
    (b) `x = 4`
    (c) `'a'`
    (d) `1`
    (e) `'a' < 'z'`
    (f) `19`
    (g) All of the above are true values.

22. Which variables are in scope at the comment?

```
int x = 6;
for (int i = 0; i < 5; ++i) {
  char c = 'a' + x + i;
}
// Here
```

    (a) `x` and `i`
    (b) The code will not compile.
    (c) `x`
    (d) `x`, `i`, and `c`
    (e) None of the variables are in scope.

23. Depending on context, what can the `*` character mean?
    (a) Multiplication Operator
    (b) De-reference Operator
    (c) Pointer Declaration
    (d) Extraction Operator
    (e) (a), (b), (c) and (d)
    (f) (a), (b) and (c)
    (g) (a) and (b)

24. Which of the following are **NOT** legal, reference initializations?
    (a) `string & x = string("Hi");`
    (b) `string & x;`
    (c) `string & x = "Hi";`
    (d) `string & x = 4;`
    (e) (b) and (c) are both not legal.
    (f) All of the above are legal.
    (g) None of the above are legal.

25. Which of the lines indicates that the pointer (`unsigned * x;`) does not point at a valid object?
    (a) `x = -1;`
    (b) `*x = -1;`
    (c) `x = nullptr;`
    (d) `*x = 0;`
    (e) `x = string::npos;`
    (f) None of the above.

26. Which of the following lines would cause `x` to hold the same value as that stored in the memory position `0x01a`?
    (a) `int x = 0x01a;`
    (b) `int y = 0x01a; int x = &y;`
    (c) `int * x = 0x01a;`
    (d) `int * y = 0x01a; int x = *y;`
    (e) `int * y = 0x01a; int & x = y;`
    (f) None of the above.

27. What is the reason to care about `const`-correctness?
    (a) Without it, references and pointers would be impossible.
    (b) It allows you to guarantee that a value can't be changed after initialization.
    (c) It allows for run-time assertions to be checked.
    (d) It converts compile-time errors into run-time errors.
    (e) None of the above.

28. If you use the Address-Of operator on a pointer to a string, what type is returned?
    (a) A const string
    (b) A pointer to a const string
    (c) A string
    (d) A pointer to a pointer to a string
    (e) A pointer to a string
    (f) None of the above

29. Which of the following statements will NOT cause a for loop to terminate?

    (a) break
    (b) return
    (c) continue
    (d) All of the above will terminate a for loop.

30. If curly braces {} aren't used to bound the body of a flow control statement (like `if`), what does that imply about the body?

    (a) That it will generate a compiler warning
    (b) That it must end in a semicolon
    (c) That it must be indented
    (d) That it is only a single statement long
    (e) All of the above are true

31. What is the output from the following code fragment?

```
int x = 7;
if (x) {
    int y = 7;
} else {
    int y = 0;
}
std::cout << x << "," << y <<
std::endl;
```

    (a) `0,7`
    (b) `7,0`
    (c) `0,0`
    (d) `7,7`
    (e) Nothing: it is illegal C++ code

32. What is the value of `z` after the following code executes?

```
int x=3, y=4;
int z = x*x+y*y;
```

    (a) 144
    (b) 12
    (c) 25
    (d) 52
    (e) 84
    (f) Nothing: it is illegal C++ code

33. In the following code, what is the final value of the variable `i`?

```
int i;
for (i=1; i < 20; i += 3) i++;
```

    (a) 1
    (b) 19
    (c) 21
    (d) 18
    (e) 22
    (f) Undefined because `i` was never initialized.

34. What will the following code print?

```
int counter = 0;
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 5; j++) {
        if (i == j) break;
        counter++;
    }
}
std::cout << counter << std::endl;
```

    (a) `20`
    (b) `12`
    (c) `10`
    (d) `6`
    (e) `0`

35. What is a difference between pointers and references in C++?

    (a) Pointers can be null (not point to anything), while references must always refer to something
    (b) References can be passed into functions, while pointers cannot
    (c) Pointers can change the value of an object, while references cannot
    (d) Pointers can be invalid if the original object is deleted, while references will always be valid
    (e) There are no differences beyond syntax

36. What will the following code print?

```
std::string word = "TEST";
for (auto & letter : word) {
    letter = letter - 'A' + 'a';
}
std::cout << word << std::endl;
```

(a) `TEST`
(b) `test`
(c) `uftu`
(d) `t`
(e) Nothing; you cannot perform math on characters.

37. Which of the following can cause the pointer named `ptr` to point at a different object in memory.

(a) `x = *ptr;`
(b) `x = ptr;`
(c) `*ptr = x;`
(d) `ptr = x;`

38. A common idiom for reading in values from standard input is used below. When will this loop terminate?

```
char c;
while (cin >> c) {
        ...
}
```

(a) Never, the loop will run forever
(b) When punctuation is encountered
(c) When a integer or floating point value is encountered
(d) When the End-Of-File is encountered
(e) When whitespace is encountered

39. When must you use curly braces ({}) in an if statement?

(a) When the if statement is nested in another flow control statement.
(b) When there is an else statement.
(c) When the body consists of multiple statements.
(d) Curly braces are never required.
(e) When there is a potential for an dangling else.
(f) Curly braces are always required

40. Which of the following permits the `x` identifier to be used to alter the value of `y`?

(a) `int const * x = &y;`
(b) None of the above allow `x` to alter `y`.
(c) `int const x = y;`
(d) `int * const x = &y;`
(e) `int const & x = y;`

41. Is the keyword `continue` allowed in an else statement?

(a) No, but `break` can be used instead for the same effect.
(b) Yes, always
(c) Yes, but only when that else statement is also inside of a loop statement
(d) No, it is a compile time error

42. If you want a newline between ”one” and ”two” in a string literal, how would you represent it?

(a) `"one" "two"`
(b) `"one" ... "two"`
(c) `"one\ntwo"`
(d) `"one" + std::endl + "two"`

43. What can the user input to output ”A”?

```
int x;
cin >> x;
if (x == (3 || 4)) {
        cout << "A";
} else {
        cout << "B";
}
```

(a) 1
(b) 2
(c) 3
(d) All of the above
(e) None of the above

44. Fall-through means that which of the following is occurring?

    (a) The switch statement does not compile unless there is a break in each case.
    (b) The switch statement will execute code in the case and all following cases until a break or the end of the switch statement.
    (c) That the default case will always run after the matching case completes.
    (d) The switch statement executes the code in the matching case, and then immediately exits the switch statement regardless of cases afterwards.

45. In C++, a variable's scope determines what about it?

    (a) The region of the code that is able to read/write to that variable.
    (b) The amount of memory allocated for the variable.
    (c) The duration during which the variable is valid.
    (d) The range of possible values that the variable can take on.

46. How many integers would the following program print?

```
int x = 6;
while (x > 0) {
        cout << x << endl;
        x -= 2;
        if (x == 2) {
                break;
        }
}
```

    (a) 0 (nothing is printed)
    (b) 1
    (c) 2
    (d) 3
    (e) 4
    (f) 5
    (g) 6

47. How many integers would the following program print?

```
for (int x = 0; x < 4; ++x) {
        if (x == 2) {
                continue;
        }
        cout << x << endl;
}
```

    (a) 0 (nothing is printed)
    (b) 1
    (c) 2
    (d) 3
    (e) 4
    (f) 5
    (g) 6

48. What is the output from the following code?

```
double x = 1.05;
double y = 3.15;
double z = y / x;
if (x < z) std::cout << "ONE ";
if (y < z) std::cout << "TWO ";
else if (x+y > z) std::cout <<
"THREE ";
```

    (a) `TWO`
    (b) `TWO THREE`
    (c) `ONE THREE`
    (d) `THREE`
    (e) `ONE`
    (f) `ONE TWO`
    (g) `ONE TWO THREE`

49. What is the name of the ampersand (`&`) operator?

    (a) None of the above
    (b) The dereference operator
    (c) The and operator
    (d) The address-of operator
    (e) The pointer operator

50. What is `x` in this declaration?:
`const string * x;`

    (a) A pointer to a string
    (b) A pointer to a constant string
    (c) None of the above
    (d) A constant pointer to a constant string
    (e) A constant pointer to a string
    (f) Syntax Error

51. Can you declare (without initialization) a reference?

    (a) No
    (b) Depends on if the reference is const
    (c) Depends on if the reference is for a fundamental type

52. How do you stop fall-through behavior?

    (a) By ensuring that you have a break at the end of each case
    (b) By using a conditional statement to check for const expressions
    (c) By ensuring that you have a default case
    (d) You can't stop fall-through, it is mandated by the C++ language

53. When you increment a pointer, for instance:
`++pointer_variable;`
What happens?

    (a) The pointer now points to the next address in memory
    (b) Nothing happens, because the prefix increment was used, only the value returned is affected
    (c) A syntax error is thrown by the compiler as pointers can't be incremented
    (d) Undefined behavior, the language specification doesn't specify what will happen, so the result is undefined
    (e) The value at the pointer's address is incremented by one

54. How many iterations of the while loop will occur?

```
while (7) {
    int x = 4;
    ++x;
    if (x > 6) {
        break;
    }
}
```

    (a) 0 iterations
    (b) 1 iteration
    (c) 2 iterations
    (d) 4 iterations
    (e) 6 iterations
    (f) 7 iterations
    (g) more than 13 iterations

55. What is a pointer's value?

    (a) A signed long
    (b) A const reference to another object
    (c) An address in memory
    (d) The value `nullptr` unless it was initialized or assigned
    (e) All of the above

56. The `continue` statement is different from the `break` statement in one key way, what is it?

    (a) It always creates an infinite loops, unless a `break` statement is also included.
    (b) It is only supported by specific compilers and isn't in the language standard
    (c) It is only permitted in while loops, but not in other iterative statements.
    (d) It is actually identical to `break`, its use is entirely aesthetic.
    (e) It can't be used in blocks due to the ambiguity.
    (f) It requires the use of an `if` statement.
    (g) It causes iteration to resume instead of cease.
    (h) Its use is strongly discouraged due to the poor habits it inspires

57. After the code below executes, which pointers have the same value as `a`?

```
int x = 67; int y = 34;
int * a = &x;
int * b = &y;
int * c = b;
int * d = &y;
*c = 67;
*b = *a;
d = a;
*d = 34;
```

(a) `b`
(b) `c`
(c) `d`
(d) Both `b` and `c`
(e) All of `b`, `c`, and `d`
(f) None of them have the same value as `a`
(g) The code is invalid, and thus no answer can be given

58. Declaring all local variables at the beginning of a function is bad practice because it results in unnecessarily large XXXXs. What term should replace XXXXs?

(a) Scopes
(b) Code sizes
(c) Memory uses
(d) Functions
(e) Blocks
(f) Exceptions
(g) Variable names
(h) Comments

59. Which type of pointer should you NOT dereference?

(a) A pointer with a memory address
(b) A pointer that points at another pointer
(c) A null pointer
(d) A pointer to a local variable
(e) A const pointer
(f) A pointer to a null character
(g) A pointer to a const object
(h) A pointer that has been assigned

60. Which of the following can you NOT make a const reference to?

(a) A non-const value
(b) A literal value
(c) A pointer
(d) A value returned by a function
(e) A reference
(f) A const value
(g) All of the above permit const references

61. How do you create a null reference?

(a) By assigning it the value `0`
(b) By using `const_cast`
(c) By allowing the reference's lifetime to end
(d) By assigning it the value `false`
(e) By making a reference to a dereferenced null pointer
(f) By subtracting one from a null character
(g) None of the above, it is impossible

62. For what values of `x` will this program have a 'b' character in the output?

```
switch (x + 1) {
  case 0:
  case 1:
    std::cout << 'a';
  case 2:
    std::cout << 'b';
  default:
    std::cout << 'c';
}
```

(a) `-1`
(b) `0`
(c) `1`
(d) `2`
(e) `3`
(f) `-1`, `0`, and `1`
(g) `0`, `1`, and `2`
(h) `-1`, `1`, `2`, and `3`
(i) All possible values of `x`

63. When will the body of the if statement execute?

```
if (x || y) {
  ...
}
```

  (a) Only when `x` is true
  (b) Only when `y` is true
  (c) Only when `x` and `y` are both true
  (d) Only when one of either `x` or `y` are true
  (e) Only when `x` and/or `y` are true
  (f) Impossible to determine with the information given

64. If you use the Address-Of operator on a pointer to a string, what type is returned?

  (a) A pointer to a pointer to a string
  (b) A string
  (c) A pointer to a string
  (d) A const string
  (e) A pointer to a const string
  (f) None of the above

65. Which of the following statements will NOT cause a for loop to terminate?

  (a) break
  (b) return
  (c) continue
  (d) All of the above will terminate a for loop.

66. If curly braces {} aren't used to bound the body of a flow control statement (like `if`), what does that imply about the body?

  (a) That it must be indented
  (b) That it will generate a compiler warning
  (c) That it must end in a semicolon
  (d) That it is only a single statement long
  (e) All of the above are true

67. What is the type of `x`?

```
string const s = "hi";
string const * const ptr_s = &s;
auto y = *ptr_s;
auto x = &y;
```

  (a) `string const &`
  (b) `string`
  (c) `string const`
  (d) `string const *`
  (e) `string const * const`
  (f) `string *`
  (g) `string &`
  (h) None of the above.

68. Which of the following are illegal to have two of?

  (a) Two references to the same variable
  (b) Two pointers to the same object
  (c) Two includes of the same header
  (d) Two functions with the same name
  (e) None of the above are illegal

69. Presuming `x` is a pointer, the expression `*x` will return which of the following?

  (a) The address of the value of `x`
  (b) The value of `x`, if `x` is a pointer its address, otherwise its value
  (c) The value at the address held by `x`
  (d) The address of `x`
  (e) The type of `x`
  (f) The value of `x`
  (g) None of the above

70. If `char` variable named `c` is declared, what is the type of the expression `&c`?

  (a) `char *`
  (b) `char &`
  (c) `char ptr`
  (d) `& char`
  (e) `char`
  (f) None of the above

71. After initializing a reference, how do you change what it is referring to?

    (a) By changing its address
    (b) By using a cast
    (c) By calling a function
    (d) By assigning to it
    (e) By using static typing
    (f) None of the above

72. How do you avoid copying an object during a function call or in a range-for-loop?

    (a) By using references
    (b) By ensuring that the object is never changed
    (c) By marking it const
    (d) By naming it in all capital letters
    (e) By declaring a new variable
    (f) It is impossible

73. If a pointer has the value `nullptr`, what does that mean about the pointer?

    (a) The pointer shouldn't be used with the unary `*`
    (b) The pointer needs to be incremented
    (c) The pointer is const
    (d) The pointer is pointing at the end of a string
    (e) The pointer hasn't been assigned a value yet
    (f) None of the above

74. When is the `default` case in a switch statement executed?

    (a) When multiple cases match the value
    (b) After all the other cases execute
    (c) When no value is provided
    (d) When the value has a false/zero value
    (e) When no other case matches the value
    (f) None of the above

75. When used in a condition of an if-statement, which of the following values are considered true?

    (a) `0`
    (b) `false`
    (c) `nullptr`
    (d) `4 != 0`
    (e) `7 > 8 > 9`
    (f) None of the above

76. After the following statements, which of the following expressions would yield the value 232?

```
int a = 232;
int * b = &a;
int & c = a;
```

    (a) `*a`
    (b) `&b`
    (c) `&a`
    (d) `&c`
    (e) `*b`
    (f) None of the above

77. Which of the following is an assignment (not an initialization)?

    (a) `char c = '!';`
    (b) `int i{2};`
    (c) `int array[] = {1, 2, 3};`
    (d) `vec[4] = "";`
    (e) `double & d = 4.3;`
    (f) None of the above are assignments

78. Which of the following statements are true about C-style strings?

    (a) `"word"` is an example of a C-style-string literal
    (b) It is an array of char
    (c) They are different from `std::string`
    (d) The last character is always the null character
    (e) All of the above are true

79. After the following code, which of the following statements would generate compile-time errors?

```
double score = 98.6;
double const * grade = &score;
```

(a) `grade = nullptr;`
(b) `score += 6.5;`
(c) `*grade = 0.0;`
(d) `double other = 0;`
    `grade = &other;`
(e) Multiple previous options would generate compile-time errors

80. What does the following code output?

```
char c = 'D';
if (c == 'd')
  cout << 4;
  cout << '5';
cout << "6";
```

(a) Nothing is output
(b) `4`
(c) `45`
(d) `456`
(e) `56`
(f) `6`
(g) `"6"`
(h) The above code wouldn't compile

81. In the following code, what expression is executed after the `continue` is executed?

```
for (int x = 0; x < 5; ++x) {
  if (x % 3 == 0) {
    cout << "x is divisible"
      << endl;
    continue;
  }
  cout << "x is " << x << endl;
}
```

(a) `cout << "x is divisible"`
(b) `x = 0`
(c) `++x`
(d) `x < 5`
(e) `cout << "x is " << x << endl;`
(f) The above code could wouldn't compile

82. What is the difference between these two loops?

```
for (int x = 0; x < 5; ++x) {
   \\Loop Body Omitted
}
```

and

```
int x = 0;
while (x < 5) {
   \\Loop Body Omitted
   ++x;
}
```

(a) The number of iterations is different
(b) The final value of `x` is different
(c) The scope of `x` is different
(d) There is no difference

83. When will the following expression result in a false value?

```
cin >> x
```

(a) When the get from (`>>`) operation fails due to no more input being provided
(b) When the get from (`>>`) operation fails due an invalid provided input
(c) When the End-Of-File character is encountered
(d) Multiple of the above are true

84. If variable `c` is declared const, like as shown below, which of the following statements would result in an error when the program was executed?

```
char const c = 'A';
```

(a) `c = 'B';`
(b) `cout << c;`
(c) `c++;`
(d) `char c2{c};`
(e) `c += 'D';`
(f) None of the above would result in a run-time error.

85. What does the following code output?
```
int x = 9;
while (4 = x) {
  cout << x;
  cout << x++;
  if (7) break;
  cout << x;
}
cout << x;
```

   (a) `445`
   (b) `4`
   (c) `9`
   (d) `455`
   (e) `9101010`
   (f) `991010`
   (g) It doesn't compile.
   (h) It never ends.

86. A variable has the value `0x123ace`, what is its type?

   (a) A pointer
   (b) `char`
   (c) `double`
   (d) `int`
   (e) Impossible to determine with the information given

87. What does the following statement do?
```
cin >> std::noskipws
```

   (a) It causes `std::cin` to count the number of characters that are read from standard input.
   (b) It sets a flag in `std::cin` to cause it to no longer skip flushing the the stream when a newline character is encountered.
   (c) It does nothing, but allows the code to compile and run on Unix environments.
   (d) It indicates that `std::cin` should only provided one character at a time, except for characters that are word size or smaller.
   (e) It causes `std::cin` to not ignore whitespace characters when using the get from operator (`>>`).

88. Why is the following code illegal?
```
int x = 7;
if (x) {
    int y = 7;
} else {
    int y = 0;
}
std::cout << x << "," << y <<
std::endl;
```

   (a) Variable `y` is declared multiple times.
   (b) Variable `y` is accessed outside of its scope.
   (c) Variable `y` is being assigned multiple values, but a variable can only hold one value at a time.
   (d) The if-statement's conditional isn't checking anything.
   (e) The code above isn't illegal.

89. What is the difference between these two while loops?
```
while(true) {
  \\...
}
```
and
```
while(7) {
  \\...
}
```

   (a) The first loop will execute one time, the second will iterate 7 times.
   (b) The first loop will execute an unlimited number of times, the second only 7 times.
   (c) The second loop won't compile as `7` isn't a bool value.
   (d) The first loop can't be terminated with a `break` statement, the second one can.
   (e) The loops are functionally the same.

This page intentionally left blank.