Week 02 Sample Exam

CSE 232 (Introduction to Programming II)

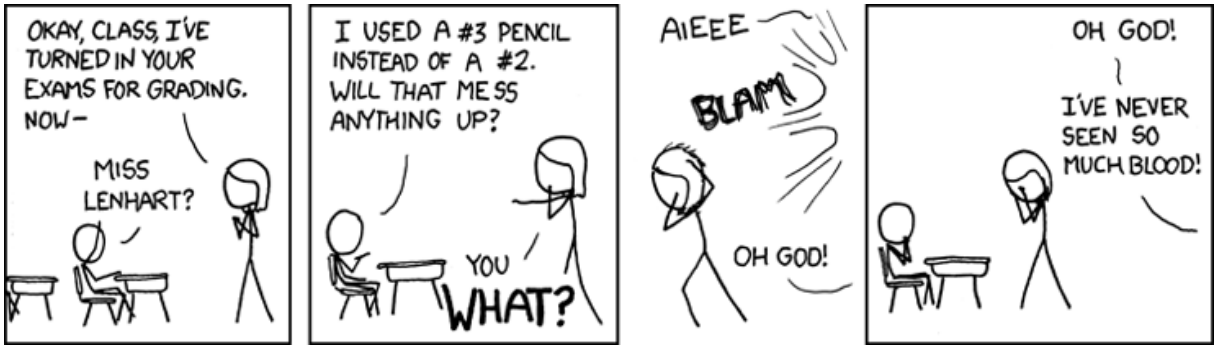
VERSION A

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 LETTER** (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). Please be sure to bring the required textbook!
- If you have any questions during the exam or when you finish the exam, please raise your hand and a proctor will attend you.



http://xkcd.com/499/ Date Accessed: October 16, 2024

1. What does the following code print?
```
int a{20};
int * x{&a};
cout << x << '\n';
```

   (a) `20`
   (b) A memory address
   (c) Nothing, code will not compile
   (d) `a\n`
   (e) Nothing, a segmentation fault occurs
   (f) `x\n`

2. 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.

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

4. How do you stop fall-through behavior?

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

5. 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.

6. 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.

7. 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.

8. 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.

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

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

10. 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

11. Why can you not declare a reference?

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

12. 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, but only pointers that aren't null
   (d) Yes, pointers like all objects, have addresses

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

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

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

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

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

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

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

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

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

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

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

    (a) `;`
    (b) `"`
    (c) `'`
    (d) `)`
    (e) `>`
    (f) `]`
    (g) None of the previous.

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

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

20. 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.

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

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

22. Which of the following expressions do **NOT** evaluate to a true value?

    (a) `1`
    (b) `true`
    (c) `'a' < 'z'`
    (d) `19`
    (e) `'a'`
    (f) `x = 4`
    (g) All of the above are true values.

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

24. 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)

25. 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.

26. Which of the lines indicates that the pointer (`unsigned * x;`) does not point at a valid object?

    (a) `x = -1;`
    (b) `x = nullptr;`
    (c) `*x = -1;`
    (d) `*x = 0;`
    (e) `x = string::npos;`
    (f) None of the above.

27. 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 * y = 0x01a; int & x = y;`
    (d) `int x = 0x01a;`
    (e) `int * y = 0x01a; int x = *y;`
    (f) None of the above.

28. What is the reason to care about `const`-correctness?

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

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

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

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

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

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

32. 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) `7,7`
    (c) `7,0`
    (d) `0,0`
    (e) Nothing: it is illegal C++ code

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

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

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

34. 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) 18
    (d) 21
    (e) 22
    (f) Undefined because `i` was never initialized.

35. 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`

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

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

37. What will the following code print?

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

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

38. 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;`

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

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

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

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

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

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

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

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

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

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

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

44. 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

45. 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) That the default case will always run after the matching case completes.
    (c) The switch statement executes the code in the matching case, and then immediately exits the switch statement regardless of cases afterwards.
    (d) The switch statement will execute code in the case and all following cases until a break or the end of the switch statement.

46. 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 duration during which the variable is valid.
    (c) The amount of memory allocated for the variable.
    (d) The range of possible values that the variable can take on.

47. 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

48. 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

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

50. 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

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

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

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

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

53. How do you stop fall-through behavior?

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

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

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

55. 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

56. What is a pointer's value?

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

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

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

58. 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

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

60. Which type of pointer should you NOT dereference?

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

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

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

62. How do you create a null reference?

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

63. 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`

64. 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

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

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

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

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

67. 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 is only a single statement long
(d) That it must end in a semicolon
(e) All of the above are true

68. 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`
(b) `string *`
(c) `string &`
(d) `string const`
(e) `string const *`
(f) `string const &`
(g) `string const * const`
(h) None of the above.

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

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

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

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

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

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

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

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

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

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

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

    (a) The pointer is const
    (b) The pointer shouldn't be used with the unary `*`
    (c) The pointer needs to be incremented
    (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

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

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

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

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

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

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

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

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)
    ```
    double other = 0;
    grade = &other;
    ```
    (c) `score += 6.5;`
    (d) `*grade = 0.0;`
    (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 " << x << endl;`
(b) `cout << "x is divisible"`
(c) `++x`
(d) `x < 5`
(e) `x = 0`
(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 scope of `x` is different
(c) The final value 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 an invalid provided input
(b) When the End-Of-File character is encountered
(c) When the get from (>>) operation fails due to no more input being provided
(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) `char c2{c};`
(c) `c += 'D';`
(d) `cout << c;`
(e) `c = 'B';`
(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) `double`
(b) `int`
(c) A pointer
(d) `char`
(e) Impossible to determine with the information given

87. What does the following statement do?

```
cin >> std::noskipws
```

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

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) The if-statement's conditional isn't checking anything.
(b) Variable `y` is accessed outside of its scope.
(c) Variable `y` is declared multiple times.
(d) Variable `y` is being assigned multiple values, but a variable can only hold one value at a time.
(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 can't be terminated with a `break` statement, the second one can.
(b) The first loop will execute one time, the second will iterate 7 times.
(c) The first loop will execute an unlimited number of times, the second only 7 times.
(d) The second loop won't compile as 7 isn't a bool value.
(e) The loops are functionally the same.

90. If you omit the conditional clause of a for loop statement, which other situation would cause that for loop to terminate?

    (a) If a continue statement was executed.
    (b) If a break statement was executed.
    (c) If an exit statement was executed.
    (d) If an end statement was executed.
    (e) 2 of the above are true
    (f) 3 of the above are true
    (g) 4 of the above are true

91. What is the type of x?

```
Thing a{4};
auto b = &a;
auto c = *b;
auto x = c;
```

    (a) `auto`
    (b) `Thing & *`
    (c) `* Thing`
    (d) `& Thing`
    (e) `Thing &`
    (f) `Thing`
    (g) `int`
    (h) `Thing *`

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

```
int x = 99; int y = 88;
int * a = &y;
int * b = &y;
int * c = b;
int * d = &x;
*c = 4;
*b = *a;
d = b;
*d = 11;
```

    (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

93. Which of the following statements would generate a compile time error if included after the following code?

```
int x = 99; int y = 99;
int * a = &x;
int * const b = &y;
```

    (a) `a = b;`
    (b) `b = a;`
    (c) `*a = *b`
    (d) `*b = *a`
    (e) 2 of the above would cause errors
    (f) 3 of the above would cause errors
    (g) 4 of the above would cause errors
    (h) None of the above would cause errors

94. For the last code block in the required textbook's Section 1.7.1, what character causes the while loop to terminate?

    (a) Control-C
    (b) `' '`
    (c) `'0'`
    (d) The EndOfFile character
    (e) `'\n'`
    (f) `'\0'`
    (g) None of the above

95. Which of the following statements is TRUE?

    (a) References need only be initialized if they are const.
    (b) A const pointer can be made to point to a different location.
    (c) A pointer to a const variable can change the variable it points to, but not change which variable it points at.
    (d) A pointer to a pointer has a value that is not an address.
    (e) A reference and a pointer can both be made to access the same object.
    (f) None of the above are true.

96. What will be the output of the following code?
```
for (int i{0}; i < 4; ++i)
   cout << i;
   cout << ",";
```

(a) `0,1,2,3,4,`
(b) `1,2,3,`
(c) `1,2,3,4,`
(d) `0,1,2,3,`
(e) None of the above

97. When will the body of this if statement execute, assuming v is a string?
```
if (auto x = v[0]) { // ...
```

(a) If `x` is equal to `v`
(b) If `x` is `char`
(c) If the first value of `v` is not the null character
(d) If `v` has size greater than 1
(e) If `x` is a reference to `v[0]`
(f) If `x` is not `'0'`
(g) None of the above

98. What does a `continue` statement do when used in a case of a switch?

(a) It doesn't affect the switch statement directly, but will cause the next iteration of the loop statement.
(b) It causes the next case to execute immediately (fall-though).
(c) It is an error, `continue` statements can't be used within cases.
(d) It immediately executes the default case.
(e) It jumps to the switch statement's conditional clause.
(f) None of the above.

99. Which of the following will make a reference named `x` refer to a variable named `y`.

(a) `x = *y;`
(b) `x = y;`
(c) `*x = *y;`
(d) `*x = &y;`
(e) `x = &y;`
(f) `*x = y;`
(g) None of the above

100. For which of the following values of `x` will this program have a 'b' character in the output?
```
switch (x / 2) {
  case 1:
    std::cout << 'a';
  case 2:
    std::cout << 'b';
  default:
    std::cout << 'c';
}
```

(a) 1, 2, 3, 4
(b) 1, 2, 3, 4, 5
(c) 2, 3, 4
(d) 0, 1, 2, 3, 4
(e) 4
(f) 0, 1, 2, 3, 4, 5
(g) 4, 5
(h) All non-negative integers
(i) All possible integers
(j) 2, 3, 4, 5

101. Which of the following types support the address of operator?

(a) pointer to pointer to a `char`
(b) reference to a `bool`
(c) `int`
(d) pointer to a `double`
(e) array of `int`s
(f) All of the above

102. How many lines of output are printed by the following code?
```
for (int num = 1; ; ++num) {
   cout << num << endl;
   if (++num > 5) {
     break;
}}
```

(a) The code doesn't compile
(b) 0
(c) 1
(d) 2
(e) 3
(f) 4
(g) 5
(h) 6
(i) 7

103. Which of the following expressions results in a `true` value only if the `int` named `x` has the value of 3, 4, or 5?

(a) `x >= 3 && x <= 5`
(b) `3 <= x <= 5`
(c) `x == 3 | 4 | 5`
(d) `x <=> {3, 4, 5}`
(e) (a) and (b)
(f) (c) and (d)
(g) All of the above
(h) None of the above

104. Which of the following correctly initializes a type of `char const &` named x?

(a) `char const & x = 'a';`
(b) `char c = 'a'; char const & x = c;`
(c) `char const c = 'a'; char const & x = c;`
(d) `char c = 'a'; char const & x; x = c;`
(e) (a) and (b)
(f) (b) and (c)
(g) (a), (b), and (c)
(h) None of the above

105. Which of the following initialize a constant pointer to a double named var?

(a) `const * var double{nullptr};`
(b) `double const * var{nullptr};`
(c) `double * const var{nullptr};`
(d) `var const double *{nullptr};`
(e) `const * double var{nullptr};`
(f) `const var double *{nullptr};`

106. If a variable has the value of `nullptr`, what does that imply?

(a) That it is const
(b) That it hasn't been initialized
(c) That it is no longer being used
(d) That it points to a value of 0
(e) That is doesn't point at valid memory
(f) That it should be dereferenced before it is assigned
(g) None of the above

107. When will `b` have a `false` value?
```
char * x;
//...
bool b{x};
```

(a) When `x` is uninitialized
(b) When `x` points at the value `'0'`
(c) When `x`'s value is `nullptr`
(d) When `x` does NOT points at the value `'0'`
(e) When `x` is initialized
(f) When `x` is dereferenced
(g) When `x` is NOT dereferenced
(h) When `x`'s value is NOT `nullptr`
(i) Two of the above are true

108. How do you change the location in memory a reference refers to?

(a) Through pass by reference
(b) By assignment
(c) Through a `static_cast`
(d) Through a `const_cast`
(e) One can't; it is impossible
(f) By using the dereference operator

109. The value `0` (zero) can be stored in a variable of which of the following types?

(a) `char`
(b) `int`
(c) `char *`
(d) `double`
(e) `bool`
(f) All of the above

110. Which of the following variables have the same value after the following code runs?

```
double apple{7.8};
double banana{apple};
double carrot{7.8};
double * danish{&apple};
double * eggplant{&banana};
double * fig{&carrot};
banana = 11.4;
*fig = 11.4;
```

(a) `danish` and `eggplant`
(b) `banana` and `fig`
(c) `apple` and `banana`
(d) `eggplant` and `fig`
(e) `banana` and `carrot`
(f) `apple` and `banana` and `carrot`
(g) `apple` and `danish`
(h) `danish` and `eggplant` and `fig`
(i) None of the above are equal

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

```
int array[3]{1, 2, 3};
cout << array;
```

(a) `[1, 2, 3]`
(b) `3`
(c) The above code would generate a compiler error
(d) `1`
(e) An address in memory
(f) `1, 2, 3`

112. What will occur if you incremented a pointer?

(a) The object it points to will be one larger
(b) Nothing will happen because increment can't change a pointer
(c) The object it points to will be one larger, but only if that objects support the increment operator (i.e. not a string)
(d) It will now point to the next position in memory
(e) Most of the time noting will happen, but if the object pointed at is an integer then the value will increase by one.
(f) A compiler error would result as pointer types don't support operator++
(g) Doing so is undefined behavior, so it is impossible to be certain of the outcome
(h) None of the above

113. Which of the following statements would cause `x`'s value to change after the following code runs?

```
int x{12}, y{13};
int const & x1 = x;
int const * x2 = &x;
int * const x3 = &x;
```

(a) `*x3 = y;`
(b) `*x1 = y;`
(c) `x3 = y;`
(d) `*x2 = y;`
(e) `x2 = y;`
(f) `x1 = y;`
(g) None of the above

114. Why does the following code not compile?

```
if (char c = 'a')
   cin >> c;
```

(a) The if statements is an infinite loop
(b) The code is incorrectly indented
(c) The body of an if statement requires curly braces which denote the beginning and end of the block
(d) `c` is not a valid identifier for a variable
(e) `char` literals are bounded by double quotes, not single quotes
(f) Variables must be initialized before they are used in an expression
(g) The code does compile

115. Which of the following types are references?

```
int a = x;
int * b = x;
int * c = &x;
int & d = x;
int ~e = &x;
```

(a) `a`
(b) `b`
(c) `c`
(d) `d`
(e) `e`
(f) Both `b` and `c`

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

```
int x = 7;
if (!x) {
    int x;
    ++x;
} else {
    int x = 0;
    --x;
}
std::cout << x << std::endl;
```

(a) `-1`
(b) `0`
(c) `1`
(d) `6`
(e) `7`
(f) `8`
(g) Nothing: it is illegal C++ code

117. Why is it strongly recommended that you initialize your variables?

(a) Because uninitialized variables are slower to access than initialized ones
(b) Because initializing ones variables narrows their scope
(c) Because initializing ones variables ensures they have the correct type
(d) Because reading from uninitialized variables results in undefined behavior
(e) Because failing to initialize ones variables will result in compiler errors
(f) All of the above are true