Final Exam Study Guide

Due No due date **Points** 106 **Questions** 106 **Time Limit** 120 Minutes **Allowed Attempts** Unlimited

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	120 minutes	97.83 out of 106

Submitted Jul 19 at 6:06pm

Question 1

1 / 1 pts

Which library function performs an equivalent operation on C-strings?

```
string s1 = "Hello";
string s2 = "World";
s1 = s1 + s2;
```

- None of these
- `strlen()`
- `strcmp()`
- `strcpy()`
- Correct! `strcat()`

Question 2

1 / 1 pts

You can use a range-based loop on a 2D array.

- Correct! True
- False

Question 3

1 / 1 pts

Which statement displays the value 24 from the 2D array initialized here?

```
int a[2][3] =
{
   { 13, 23, 33 },
   { 14, 24, 34 }
};
```

- `cout << a[2][1];`
- None of these
- `cout << a[2][2];`
- Correct! `cout << a[1][1];`
- `cout << a[1][2];`

Question 4

1 / 1 pts

You cannot use a range-based loop on a 2D array.

- True
- Correct! False

Question 5

1 / 1 pts

Below is ***`insert()`***, a template function that works with a ***partially-filled array***. The function inserts the argument ***`e`*** into the array, in sorted order. The function returns ***`true`*** if it succeeds, ***`false`*** otherwise. The function contains an error; what is the error?

```
template <typename T>
bool insert(T* a, size_t& size, size_t MAX, T e)
{
    if (size < MAX) return false;
    size_t i = 0;
    while (i < size)
    {
        if (a[i] > e) break;
        i++;
```

```
    }
    for (j = size; j > i; j--)
        a[j] = a[j - 1];
    a[i] = e;
    size++;

    return true;
}
```

- The value is inserted into the wrong position
- Correct! If the array is full, the function overwrites memory outside the array.
- When a value is inserted, it erases one of the existing values
- The second loop should start at `i` and go up to `size`
- None of these

Question 6

1 / 1 pts

You can pass the 2D array ***`int a[3][3]`*** to the function ***`f(int a[3][], size_t n)`*** by calling ***`f(a, 3)`***.

- True
- Correct! False

Question 7

1 / 1 pts

When inserting a value into a partially-filled array, **in ascending order**, the insertion position is the index of the first value smaller than the value.

- True
- Correct! False

Question 8

1 / 1 pts

Which one of the following statements is the correct definition for a two-dimensional array of 20 rows and 2 columns of the type integer?

- Correct! `int num[20][2];`
- `int num[2, 20]`
- `int num[20, 2];`
- `int num[2][2];`
- None of these

Question 9

1 / 1 pts

You can pass the 2D array ***`int a[3][3]`*** to the function ***`f(int a[][3], size_t n)`*** by calling ***`f(a, 3)`***.

- Correct! True
- False

Question 10

1 / 1 pts

C-strings are char pointers to the first character in a sequence of characters, terminated with a `'0'` character.

- True
- Correct! False

Question 11

1 / 1 pts

When initializing a 2D, each column must have its own set of braces.

- True
- Correct! False

Question 12

1 / 1 pts

Which value of ***a*** is stored in the ***val*** variable?

```
auto val = a[0][2];
```

- The value in the third row and the first column
- The value in the first row and the second column
- None of these
- The value in the first row and the first column

Correct!

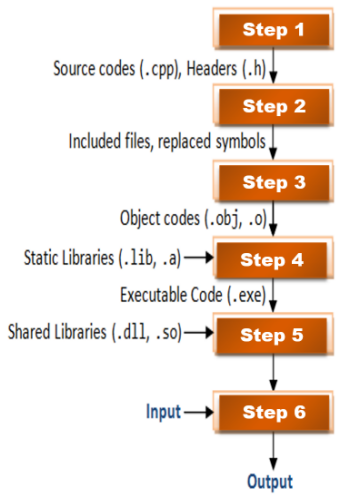
- The value in the first row and the third column

Question 13

1 / 1 pts

To the right are the six steps of the C++ development process as shown in of your reader.

Below, match each step with the tool that accepts the shown input and produces the output as shown.



Correct! **Step 1** — Text Editor

Correct! **Step 3** — Compiler

Correct! **Step 4** — Linker

Correct! **Step 5** — Loader

Other Incorrect Match Options:

- Linter
- Debugger
- Assembler

Question 14

1 / 1 pts

In the reading you learned about five concepts related to variables. Line 1 is _____. More than one answer may be required.

```
int a;              // 1
extern int b;       // 2
double c = 3.15;    // 3
a = 4;              // 4
cin >> b;           // 5
```

- input
- assignment

Correct!

- definition
- initialization

Correct!

- declaration

It is a declaration because it associates a name with a type. It is a definition because it creates an object in memory.

Question 15

1 / 1 pts

What manipulator ensures that large floating-point number appear using regular decimal notation?

- hex

- setprecision
- setw
- scientific

Correct!

- fixed
- decimal

Question 16

1 / 1 pts

In the reading you learned about five concepts related to variables. Line 4 is _____.

```
int a;              // 1
extern int b;       // 2
double c = 3.15;    // 3
a = 4;              // 4
cin >> b;           // 5
```

Correct!

- assignment
- input
- initialization
- declaration
- definition

It stores a value in an existing variable.

Question 17

0 / 1 pts

What happens here?

```
#include <iostream>
using namespace std;
int main()
{
    int size = 42;
    cost = 9.99;
    cout << "size=" << size
        << ", cost=" << cost << endl;
}
```

You Answered

- Prints size=42, cost=9.99
- Logic error, uninitialzed variable.
- Prints size=42, cost=9
- Syntax error: assign double to int

Correct Answer

- Syntax error: undeclared variable

Question 18

0 / 1 pts

Used by compiler to produce object code.

- Debugger

You Answered

- Compiler
- Preprocessor
- Loader
- Linker

Correct Answer

- Assembler
- Make

Question 19

1 / 1 pts

What is printed here?

Correct!

```
#include <iostream>
using namespace std;
int main()
{
    int n = 2;
    cout << ++++n << endl;
}
```

Question 20

1 / 1 pts

Which of the following is the correct syntax for an `if-else` statement?

Correct!

```
--
if (x < 10)
{
    size = "Small";
}
else
{
    size = "Medium";
}
```

```
--
if (x < 10)
{
    size = "Small";
}
else (x < 20)
{
    size = "Medium";
}
```

```
--
if
{
    size = "Small";
}
else (x < 20)
{
    size = "Medium";
}
```

```
--
if (x < 10);
{
    size = "Small";
}
else (x < 20)
{
    size = "Medium";
}
```

Question 21

1 / 1 pts

Which condition, when supplied in the `if` statement below in place of `(. . .)`, will correctly protect against division by zero?

```
if (. . .)
{
   result = grade / num;
   cout << "Just avoided division by zero!" << endl;
}
```

```
(grade == 0)
```

```
((grade / num) == 0)
```

Correct!

```
(num != 0)
```

```
(num == 0)
```

Question 22

1 / 1 pts

Assuming that a user enters 45, 78, and 12 one after another, separated by spaces, what is the output of the following code snippet?

```
int num1, num2, num3 = 0;

cout << "Enter a number: ";
cin >> num1;

cout << "Enter a number: ";
cin >> num2;

cout << "Enter a number: ";
cin >> num3;

if (!(num1 > num2 && num1 > num3))
{
    cout << num1 << endl;
}
else if (!(num2 > num1 && num2 > num3))
{
    cout << num2 << endl;
}
else if (!(num3 > num1 && num3 > num2))
{
    cout << num3 << endl;
```

```
}
```

- 12
- Correct! 45
- 78
- There is no output due to compilation errors.

Question 23

1 / 1 pts

What is printed when this runs?

```
#include <iostream>
using namespace std;
int main()
{
    int a = 3, b = ++a;
    cout << "a->" << a << ", b->" << b << endl;
}
```

- a->4, b->3
- Anything, this is undefined beahvior.
- Correct! a->4, b->4
- This is a syntax error.

Preincrement in the initialization of b.

Question 24

1 / 1 pts

What is printed here?

```
#include <iostream>
using namespace std;
int main()
{
    int x = 20, y = 10;
    y = x--;
    cout << x << ", " << y << endl;
}
```

- Correct! 19, 20
- 20, 19
- 19, 19
- 20, 20

Question 25

1 / 1 pts

What is the output of the following code snippet?

```
double income = 45000;
double cutoff = 55000;
double min_income = 30000;

if (min_income > income)
{
    cout << "Minimum income requirement is not met." << endl;
}
if (cutoff < income)
{
    cout << "Maximum income limit is exceeded." << endl;
}
else
{
    cout << "Income requirement is met." << endl;
}
```

- There is no output.
- Minimum income requirement is not met.
- Maximum income limit is exceeded.
- Correct! Income requirement is met.

Question 26

0.5 / 1 pts

If a and c are both `false`, which expressions **need not** be evaluated?

```
1 | if (a && b || c && d || e) . . .
```

Correct! ☑ b

Correct! ☑ d

☐ c

You Answered ☑ e

☐ a

Question 27

1 / 1 pts

This loop uses asymmetric bounds.

```
for (int i = 0; i < 10; i++)
    cout << i;
cout << endl;
```

Correct! ◉ True

○ False

Question 28

1 / 1 pts

What prints?

```
string str = "Hello";
for (auto i = 0, len = str.size(); i < len; i++)
    cout << str.at(i);
```

○ `Hell`

○ `Hello`

○ Undefined behavior

Correct! ◉ Does not compile

○ Crashes when run

Question 29

1 / 1 pts

Look at the problem statement below. The _____ of the loop is that a period was encountered.

> *How many characters are in a sentence? Count the characters in a string until a period is encountered. If the string contains any characters, then it will contain a period. Count the period as well.*

○ None of these

○ plan

○ goal

Correct! ◉ bounds

Question 30

1 / 1 pts

Which line ***advances the loop***?

```
1.      string s("Hello CS 150");
2.      while (s.size())
3.      {
4.         if (s.at(0) == 'C') break;
5.         s = s.substr(1);
6.      }
7.      cout << s << endl;
```

○ 4

○ None of these

Correct!

- (•) 5
- ( ) 2

Question 31

1 / 1 pts

This idiomatic pattern is used to count from one value to another.

```
for (int i = 0; i < 10; i++)
    cout << i;
cout << endl;
```

- ( ) True

Correct!

- (•) False

Question 32

1 / 1 pts

Match each item with the correct statement below.

Correct! **May not repeat its actions at all** — guarded loop

Correct! **Repeats its actions at least once** — unguarded loop

Correct! **Test for the occurrence of a particular event** — indefinite loop

Correct! **Conditions under which a loop will repeat its actions** — loop bounds

Question 33

1 / 1 pts

Using the loop-building strategy from the lessons, which of these are part of the ***loop mechanics***?

- [ ] post condition

Correct!

- [x] bounds precondition
- [ ] goal precondition
- [ ] goal operation

Correct!

- [x] advancing the loop

Correct!

- [x] loop bounds

Question 34

1 / 1 pts

What kind of error is this?

```
ex1.cpp:7:9: warning: missing terminating '"' character
    a = "hello world';
        ^
ex1.cpp:7:9: error: expected expression
```

Correct!

- (•) Syntax error (mistake in grammar)
- ( ) Operating system signal or trap
- ( ) Compiler error (something is missing when compiling)
- ( ) None of these
- ( ) Runtime error (throws exception when running)
- ( ) Linker error (something is missing when linking)
- ( ) Type error (wrong initialization or assignment)

Question 35

1 / 1 pts

What prints here?

```
int i = 5;
while (i) cout << i--;
cout << endl;
```

- ( ) Syntax error: `i` is not a Boolean expression

- 4321
- 43210
- Infinite loop

Correct!

- 54321

Question 36

1 / 1 pts

An ***undefined*** error message is a compiler error.

- True

Correct!

- False

Question 37

1 / 1 pts

To allow *f()* to change the argument passed here, the parameter *str* should be declared as:

```
void f( . . . str);
int main()
{
    string s = "hello";
    f(s);
}
```

Correct!

- string&
- string
- const string
- It is not possible for *f()* to change the argument passed here.
- const string&

Question 38

1 / 1 pts

Assume that the input is `4 4 3 2 5`. What will print?

```
int i = 1;
int n;
cin >> n;
do
{
    i++;
    cin >> n;
}
while (n % 2);
cout << i << endl;
```

- 3
- Does not compile
- infinite loop
- 4

Correct!

- 2

Question 39

1 / 1 pts

A tool named Doxygen is often used to generate HTML user docs from C++ code.

Correct!

- True
- False

Question 40

1 / 1 pts

Which line in the function "skeleton" below contains an error?

```
#include "digits.h"         // 1.
int firstDigit(int n);      // 2.
{                           // 3.
    return 0;               // 4.
}                           // 5.
```

- // 5.

- None of these
- `// 1.`
- `// 3.`

Correct!

- `// 2.`
- `// 4.`

Question 41

1 / 1 pts

The return value of the `getline()` function is a `string` object.

- True

Correct!

- False

Question 42

1 / 1 pts

What does this function do?

```
int mystery(int n)
{
  if (n < 2) return 1;
  return mystery(n-1) + mystery(n-2);
}
```

- Computes the Gauss series (sum) of 1..n

Correct!

- Computes the Fibonacci number n
- Computes the reverse of the input n
- Computes the Factorial number n
- Produces a stack overflow

Question 43

1 / 1 pts

When using the `get()` member function, a stream will fail only if there are no characters left in the input stream.

Correct!

- True
- False

Question 44

1 / 1 pts

This command: `cat < nofile 2> /dev/null` will print an error message on the screen if `nofile` does not exist.

- True

Correct!

- False

Question 45

1 / 1 pts

What is the value of ***r("hello")***?

```
string r(const string& s)
{
  if (s.size() < 2) return s;
  return s.substr(0, 1) + "*" + r(s.substr(1));
}
```

Correct!

- `"h*e*l*l*o"`
- `"*h*e*l*lo"`
- Stack overflow
- `"*h*e*l*l*o"`
- `"h*e*l*l*o*"`

Question 46

1 / 1 pts

This loop:

```
char c;
while (in.get(c))
{
    cout << c << endl;
}
```

- illustrates token-based stream processing
- Correct! illustrates raw character I/O
- illustrates line-based stream processing
- is an endless loop
- has a syntax error

Question 47

1 / 1 pts

Calling `cout.put(65.35)` is illegal. Your code will not compile.

- True
- Correct! False

Question 48

1 / 1 pts

A specialized error handling block of code, is called a `try` block.

- True
- Correct! False

Question 49

1 / 1 pts

You can report a syntax error encountered in your code by using the `throw` keyword.

- True
- Correct! False

Question 50

1 / 1 pts

The order of the `catch` blocks does not affect the program.

- True
- Correct! False

Question 51

1 / 1 pts

The function ____ returns a string containing an appropriate message.

- when
- log
- where
- Correct! what

Question 52

1 / 1 pts

What prints?

```
string s("hello");
try {
    if (s.size() > 20) throw 42;
    if (isupper(s.back())) throw "goodbye";
    if (s == "Hello") throw string("hello");
    s.at(s.size()) = 'x';
    cout << "one\n";
}
catch (const int& e) { cout << "two\n"; }
catch (const string& e) { cout << "three\n"; }
catch (exception& e)  { cout << "four\n"; }
catch (...) { cout << "five\n"; }
```

- one
- two

Correct!

- four
- five
- Undefined
- three

Question 53

1 / 1 pts

Which of these are appropriate uses of the C++ `assert` facility?

- Error conditions (such as file not found)

Correct!

- Validate function arguments under the programmer's control

Correct!

- Validate assumptions about your code

Correct!

- Validate the postcondition of a calculation

Correct!

- Debugging checks
- Validate input received by your program

Question 54

1 / 1 pts

The directives `#if defined(symbol)` and `#ifdef symbol` mean, essentially, the same thing.

Correct!

- True
- False

Question 55

1 / 1 pts

What is true about this piece of code?

```
template <typename T, typename U>
T pickle(T& a, const U& b) {
    a += b;
    return b;
}

int main()
{
    int x = 42;
    auto a = pickle(x, 4.5);
    cout << a << endl;
    cout << x << endl;
}
```

Correct!

- In main, x prints 46
- In main, a prints 4.5
- In main, x prints 46.5
- This code has a syntax error.

Correct!

- In main, a prints 4

Question 56

1 / 1 pts

The statement `v.insert(v.begin(), 3)` inserts the element `3` into the `vector` `v`, overwriting the exiting element at index `0`.

- True

Correct!

- False

Question 57

1 / 1 pts

The following code is logically correct. What is the semantically correct prototype for `mystery()`?

```
vector<double> v;
mystery(v);
```

- void mystery(const vector<int>&);
- void mystery(vector<int>);
- Either mystery(const vector<int>&); or mystery(vector<int>&); could be correct.

Correct!

- void mystery(vector<int>&);
- void mystery(vector&);

Question 58

1 / 1 pts

What prints?

```
string s("hello");
try {
    if (s.size() > 20) throw 42;
    if (islower(s.back())) throw "goodbye";
    if (s == "hello") throw string("hello");
    s.at(s.size()) = 'x';
    cout << "one\n";
}
catch (const int& e) { cout << "two\n"; }
catch (const string& e) { cout << "three\n"; }
catch (exception& e)  { cout << "four\n"; }
catch (...) { cout << "five\n"; }
```

- four
- Undefined
- three

Correct!

- five
- one
- two

Question 59

1 / 1 pts

What is the purpose of the `throw` statement?

- It is used to discard erroneous input.
- It is used to detect an error situation.

Correct!

- It is used to pass control to an error handler when an error situation is detected.
- It is used to pass arguments to another method.

Question 60

1 / 1 pts

Assume `s1` and `s2` are C++ `string` objects. Which of these calls is illegal?

```
template <typename T>
void addem(T a, T b)
{
    cout << a << " + " << b << "->"
        << (a + b) << endl;
}
```

- addem(s1, s2);
- addem(3, 4)
- None of these
- addem(4.5, 5.5);

Correct!

- addem(1.5, 2);

Question 61

1 / 1 pts

When you `throw` an exception, control immediately jumps out of the current `try` block.

Correct!

- True
- False

Question 62

1 / 1 pts

Assume `s1` and `s2` are C++ `string` objects. Which of these calls is illegal?

```
template <typename T>
void addem(T a, U b)
{
    cout << a << " + " << b << "->"
        << (a + b) << endl;
}
```

- `addem(1.5, 2);`
- Correct! None of these
- `addem(4.5, 5.5);`
- `addem(3, 4)`
- `addem(s1, s2);`

Question 63

1 / 1 pts

User-defined types that contain a single value are called ***structured*** types.

- True
- Correct! False

Question 64

1 / 1 pts

Structures data members must all be of the same type.

- True
- Correct! False

Question 65

1 / 1 pts

When passing a structure variable to a function, use ***const reference*** if the function ***should not*** modify the actual argument.

- Correct! True
- False

Question 66

1 / 1 pts

What prints when this code runs?

```
enum class Coin
{
   PENNY = 1, NICKEL = 5, DIME = 10, QUARTER = 25
};

Coin c = Coin::NICKEL;
cout << static_cast<int>(c) << endl;
```

- Does not compile; Cannot assign `Coin::NICKEL` to `c`.
- Correct! `5`
- Does not compile; Missing semicolon at end of list of members.
- `2`

Question 67

1 / 1 pts

All of these are legal C++ statements; which of them uses ***indirection***?

```
int a = 3, b = 4;
```

- `int *p = &b;`
- Correct! `int x = *p;`
- `z *= a;`
- None of these use indirection.
- `int y = a * b;`

Question 68

1 / 1 pts

Types that contain objects as **elements** are called?

- enumerations
- Correct! collections
- templates
- None of these
- generics
- abstract data types

Question 69

1 / 1 pts

The variable ***buf*** is a pointer to a region of memory storing contiguous ***int*** values. (This is similar to your homework, where you had a region of memory storing ***unsigned char*** values.) The four lines shown here are legal. ***Which operation is illegal***?

```
int *p1 = buf;
const int *p2 = buf;
int * const p3 = buf;
const int * p4 const = buf;
```

- *p3 = 5;
- p2++;
- p1++;
- Correct! *p2 = 7;
- *p1 = 3;

Question 70

1 / 1 pts

Assuming that `Star` is a structure, the declaration: `vector<Star> stars(3);` creates three uninitialized `Star` objects.

- True
- Correct! False

Question 71

1 / 1 pts

Which defines a `vector` to store the salaries of ten employees?

- Correct! `vector<double> salaries(10);`
- `vector salaries(10);`
- None of these
- `vector<double> salaries[10];`
- `vector<double> salaries{10};`
- `double salaries[10];`

Question 72

1 / 1 pts

Assume that `v` contains `[1, 2, 3]`. The result of writing `cout << v.at(4);` is undefined.

- True
- Correct! False

Question 73

1 / 1 pts

What is stored in `data` after this runs?

```
vector<int> data{1, 2, 3};
data.front();
```

- []

- `[1, 2, 3, 0]`
- None of these

Correct!

- `[1, 2, 3]`
- `[1, 2]`
- `[2, 3]`

Question 74

1 / 1 pts

All of these are legal C++ statements; which of them uses the C++ ***dereferencing operator***?

```
int a = 3, b = 4;
```

- None of these use the dereferencing operator.
- `int *p = &b;`
- `z *= a;`
- `int y = a * b;`

Correct!

- `int x = *p;`

Question 75

1 / 1 pts

What is a common pointer error?

Correct!

- Using a pointer without first initializing it
- Dereferencing a pointer
- Assigning a new value to a pointer
- Setting a pointer value to `nullptr`
- Using indirection on a pointer

Question 76

1 / 1 pts

What is printed when you run this code?

```
int *p = &0;
cout << *p << endl;
```

- The word "`nullptr`"
- No compilation errors, but undefined behavior
- The address value `0`

Correct!

- No output; compiler error.
- The address value where ***p*** is stored

Question 77

1 / 1 pts

Assume the `vector` `v` contains `[1, 2, 3]`. `v.pop_back();` changes `v` to `[1, 2]`.

Correct!

- True
- False

Question 78

1 / 1 pts

What prints?

```
vector<int> v{1, 2, 3, 4, 5};
v.pop_back();
cout << v.front() << endl;
```

Correct!

- `1`
- `4`
- Nothing; run-time error.
- Nothing; compile-time error.

- 5

Question 79

1 / 1 pts

You can compare two C-strings, ***s1*** and ***s2***, by using the `==` operator.

- True

Correct!

- False

Question 80

1 / 1 pts

C-strings are character arrays that rely on a special embedded sentinel value, the character with the ASCII code `0`.

Correct!

- True
- False

Question 81

1 / 1 pts

What is the correct prototype for mystery? (It may modify the array.)

```
const int a[] = {2, 4, 6, 8};
cout << mystery(a, 4) << endl;
```

- `int mystery(int a*, size_t n);`
- `int mystery(int a, size_t n);`
- `int mystery(int[] a, size_t n);`
- `void mystery(const int a[], size_t n);`

Correct!

- `int mystery(int *a, size_t n);`

Question 82

1 / 1 pts

Which one of the following statements is the correct definition for a two-dimensional array of 20 rows and 2 columns of the type integer?

- `int num[2][2];`
- None of these
- `int num[2, 20]`

Correct!

- `int num[20][2];`
- `int num[20, 2];`

Question 83

1 / 1 pts

Below is a ***partially-filled array***. If you are appending elements to this array in a loop, what is the correct ***loop bounds*** condition?

```
const size_t MAX = 100;
double nums[MAX];
size_t size = 0;
```

- `while (MAX < size) . . .`
- None of these
- `while (size <= MAX) . . .`

Correct!

- `while (size < MAX) . . .`
- `for (size = 0; size < MAX; size++) . . .`

Question 84

0 / 1 pts

Below is a cumulative algorithm using an array and an iterator-based loop. What is printed? (Assume all includes have been added, etc.)

```
double average(const int *beg, const int *end)
```

```
{
    if (end <= beg) return 0.0 / 0.0; // nan
    double sum = 0;
    size_t count = end - beg;
    while (beg != end) sum += *beg++;
    return sum / count;
}

int main()
{
    int a[] = {2, 4, 6, 8};
    cout << average(end(a), begin(a)) << endl;
}
```

You Answered
- 5

Correct Answer
- Not a number (*NaN*)
- Does not compile
- Endless loop when run; likely crashes.
- 4

Question 85

0 / 1 pts

The function ***mystery(const int*, const int*)*** likely employs an iterator loop.

Correct Answer
- True

You Answered
- False

Question 86

1 / 1 pts

The statement ***new int{3}***; allocates an array of three integers on the heap.

- True

Correct!
- False

Question 87

1 / 1 pts

The statement ***new int{}***; is a syntax error.

- True

Correct!
- False

Question 88

1 / 1 pts

Assuming ***p*** is a pointer to a single variable allocated on the heap, the statement ***delete[] p;*** returns the allocated memory back to the operating system for reuse.

- True

Correct!
- False

Question 89

1 / 1 pts

Examine the following portion of code and then answer the questions that follow.

```
int *p1 = new int{35};  // A.
int *p2 = &(*p1);       // B.
delete p2;              // C.
cout << *p1 << endl;    // D.
delete p1;              // E.
```

Correct! **The variable p1 is created** — on the stack.

Correct! **Which line allocates a variable in the static storage area** — None of these

Correct! **Which line uses a dangling pointer?** — Line D

Correct!

Which line is a double delete?

Line E

Other Incorrect Match Options:

- on the heap
- automatic duration
- Line C and Line E
- in the static storage area
- static duration
- Line D and Line E
- Line C
- Line A and Line B
- is a syntax error.

Line A creates an initialized variable on the heap and stores its address in the pointer p1. Line B creates p2 on the stack and points it to the same variable as p1. Line C frees the object which both pointers point to. Line D uses a dangling pointer and Line E is a double delete because p2 points to the same object as p1.

Question 90

1 / 1 pts

Memory for local variables is allocated on the stack when their definitions are encountered during runtime. This is known as automatic allocation.

Correct!

- (•) True
- ( ) False

Question 91

1 / 1 pts

The variable ***p*** is located:

```
void f()
{
    int *p = new int;
}
```

- ( ) on the stack

Correct!

- (•) on the heap
- ( ) None of these
- ( ) in the static storage area

Question 92

1 / 1 pts

If the ***new*** operator cannot allocate memory, C++ throws an exception.

Correct!

- (•) True
- ( ) False

Question 93

1 / 1 pts

A class specifies the attributes of the objects it creates through the definition of embedded functions, called member functions.

- ( ) True

Correct!

- (•) False

Question 94

1 / 1 pts

The two parts of a class are a ***public*** interface and a ***private*** implementation.

Correct!

- (•) True
- ( ) False

Question 95

0 / 1 pts

If a member function is in the ***private*** section of a class, it can only be called by other member functions of the class.

Correct Answer

- ( ) True

You Answered

- (•) False

Question 96

1 / 1 pts

Objects are variables of programmer-defined types.

Correct!

- (•) True
- ( ) False

Question 97

0 / 1 pts

With classes, the ***public interface*** includes the member functions that allow clients to access object data in a safe way.

Correct Answer

- ( ) True

You Answered

- (•) False

Question 98

1 / 1 pts

The ***`Point`*** class represents ***x***,***y*** coordinates in a Cartesian plane. What is the mistake in this operator? (*Members written inline for this problem.*)

```
class Point {
    int x_{0}, y_{0};
public:
    Point(int x, int y): x_{x}, y_{y} {}
    int x() const { return x_; }
    int y() const { return y_; }
    Point& operator+=(const Point& rhs) { . . . }
};

Point& operator+(const Point& lhs, const Point& rhs) {
    return Point(lhs) += rhs;
}
```

- ( ) The `rhs` parameter should not be `const`
- ( ) There is no error; it works fine.
- ( ) The operator should not change any of its parameters

Correct!

- (•) The operator return type should not be a `Point&`.
- ( ) Does not compile; should be a member function.

Question 99

1 / 1 pts

In a constructor, objects can be initialized immediately before the opening brace of the constructor, before any other code has been run.

Correct!

- (•) True
- ( ) False

Question 100

1 / 1 pts

What prints when this code is run? (Note that ***`struct`*** is used instead of ***`class`*** only to make all members ***`public`*** and to make the code shorter).

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

struct B  { string str() const { return "B"; }};
struct D1 : public B  { virtual string str() const { return "D1"; }};
struct D2 : public B  { string str() const { return "D2"; }};
struct D3 : public D1 { string str() const { return "D3"; }};

int main() {
    B *p1(new D1), *p2(new D2), *p3(new D3);
    cout << p1->str() << p2->str() << p3->str() << endl;
}
```

- ( ) D1D2D3
- ( ) BBD3
- ( ) D1BD3

Correct!

- (•) BBB
- ( ) Does not compile

Question 101

1 / 1 pts

Consider the ***Shape*** class hierarchy, along with ***Circle***, ***Square*** and ***Star*** from your text. The ***Shape*** class is a **concrete class**.

- True

Correct!

- False

Question 102

0.33 / 1 pts

Below you'll find a C++ class hierarchy. All classes (including **Card**) are correctly and fully implemented.

```
class Hand {
    std::vector<Card> cards;
public:
    Hand() = default;
    virtual ~Hand() = default;

    void add(const Card&);
    Card get(size_t index) const;
    virtual int score() const = 0;
    virtual void sort();
};

class PokerHand : public Hand { . . . }
class BlackjackHand : public Hand { . . . }
```

The **Hand** class:

You Answered — **relationship with vector:** [ ]

Correct Answer — has-a

You Answered — **contains an abstract function** [has-a]

Correct Answer — score

Correct! — **is a derived class of** [none]

Other Incorrect Match Options:

- derived class
- PokerHand
- is-a
- concrete class
- uses-a

The Hand class is an abstract class because it has one pure-virtual (abstract) function score. It is a base class, not a derived class. It has a has-a relationship with vector, and implements the accessor member function, get.

Question 103

1 / 1 pts

It is illegal to construct an instance of an abstract class.

Correct!

- True
- False

Question 104

0 / 1 pts

Consider the ***Shape*** class hierarchy, along with ***Circle***, ***Square*** and ***Star*** from your text. The ***Shape*** class is an **abstract base class**.

Correct Answer

- True

You Answered

- False

Question 105

1 / 1 pts

The C++ facility that allows a derived class to have multiple base classes is known as interface inheritance.

- True

Correct!

- False

Question 106

1 / 1 pts

An abstract class may, but is not required to, override its pure ***virtual*** (abstract) member functions.

- True
- False

Correct!