Midterm 3 Study Guide

Due No due date **Points** 25 **Questions** 25 **Time Limit** 30 Minutes **Allowed Attempts** Unlimited

Attempt History

	Attempt	Time	Score
KEPT	Attempt 4	21 minutes	23 out of 25
LATEST	Attempt 10	17 minutes	22 out of 25
	Attempt 9	20 minutes	20 out of 25
	Attempt 8	21 minutes	20 out of 25
	Attempt 7	25 minutes	21.5 out of 25
	Attempt 6	25 minutes	21 out of 25
	Attempt 5	30 minutes	17 out of 25
	Attempt 4	21 minutes	23 out of 25
	Attempt 3	26 minutes	19.89 out of 25
	Attempt 2	30 minutes	22.5 out of 25
	Attempt 1	27 minutes	21 out of 25

ⓘ Correct answers are hidden.

Submitted Jul 13 at 3:47am

Question 1 1 / 1 pts

Which line fails to work correctly?

```
template <typename T>
void print(const T& item)
{
    cout << item << endl;
}
```

- ○ `print(2 + 2);`
- ○ `print(3 + 2.2);`
- ○ `print(string("goodbye"));`
- ● None of these
- ○ `print("hello");`

Question 2 1 / 1 pts

To use strings as a data stream source or sink, use the `<sstream>` header

- ● True
- ○ False

Question 3 1 / 1 pts

What happens when you execute the (erroneous) line:

```
auto x = sqrt(-1);
```

- ○ The statement is skipped and the program continues
- ○ The function throws and exception. If not caught the program terminates.
- ○ The code does not compile. You cannot take the square root of a negative number.
- ● The function returns an error value and the program continues
- ○ The program prints an error message and terminates

Question 4 1 / 1 pts

What header file to you need to include to use the standard C++ error-handling classes?

- ○ `<cerror>`
- ○ `<cstdlib>`
- ○ `<except>`
- ○ None of these

○ `<stderror>`

◉ `<stdexcept>`

Question 5   1 / 1 pts

A(n) _____ is an occurrence of an undesirable situation that can be detected during program execution.

○ misfire

◉ exception

○ crash

○ bug

Question 6   1 / 1 pts

The `#if` preprocessor directive may compare `double` literals but not variables.

○ True

◉ False

Question 7   1 / 1 pts

Code that may cause an error should be placed in a _____ block and code that handles the error should be inside a _____ block?

○ `throw, catch`

◉ `try, catch`

○ `raise, except`

○ `if, else`

○ `catch, try`

○ None of these

Question 8   1 / 1 pts

Without `try` and `catch`, the `throw` statement terminates the running program.

◉ True

○ False

Question 9   1 / 1 pts

In C++, any class can be considered an exception class.

◉ True

○ False

Question 10   0 / 1 pts

Structures are ***homogenous*** data types.

◉ True

○ False

Question 11   1 / 1 pts

Examine the following definition. `empID` is a _____.

```
struct Employee
{
    long empID;
    std::string lastName;
    double salary;
};
```

○ field

○ type-id

○ instance variable

○ None of these

⦿ `data member`

○ `structure tag`

Question 12   1 / 1 pts

It is ***illegal*** to include the same `struct` definition multiple times, even if the definitions are exactly the same.

⦿ True

○ False

Question 13   1 / 1 pts

The structure and variable definitions are fine. Which statements are legal?

`struct Rectangle { int length, width; } big, little;`

○ None of these are correct

⦿ `cin >> little.width;`

○ `cout << Rectangle.length;`

○ `cin >> big;`

○ `double p = 2 * (length + width);`

Question 14   1 / 1 pts

Which of these are true?

```
int main()
{
    vector<int> v{1, 2, 3};
    for (int i = v.size() - 1; i >= 0; i--)
        cout << v.at(i) << " ";
    cout << endl;
}
```

☐ Endless loop (may crash, but not necessarily)

☑ Prints `3 2 1`

☐ Issues a compiler warning, but no error

☐ Compiler error (does not compile)

☐ Crashes when run

Question 15   1 / 1 pts

User-defined types that combine multiple values into a single type are called ***structured*** types.

⦿ True

○ False

Question 16   1 / 1 pts

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

⦿ True

○ False

Question 17   1 / 1 pts

To count the number of elements in a `vector` that match a particular condition, use the STL function:

○ search

◉ count_if

○ find

○ minmax_element

○ count

Incorrect | **Question 18** | 0 / 1 pts

What is printed when you run this code?

```cpp
int *n{nullptr};
cout << n << endl;
```

○ No output; compiler error.

○ The address value 0

◉ No compilation errors, but undefined behavior

○ The word "nullptr"

Question 19 | 1 / 1 pts

What is true about this code?

```cpp
int * choice;
```

○ Syntax error; should be int choice*;

○ choice can point to any kind of object

○ choice currently points to an integer

○ choice currently contains an integer

◉ choice contains an undefined address

Question 20 | 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*?

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

◉ *p2 = 7;

○ p2++;

○ *p3 = 5;

○ p1++;

○ *p1 = 3;

Question 21 | 1 / 1 pts

Explicitly initializing an array like this: *int a[] = {1, 2, 3};* works in all versions of C++.

◉ True

○ False

Question 22 | 1 / 1 pts

Which array definition produces *{1, 2, 0}*?

```cpp
int SIZE = 3;
int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[3] = {1, 2};
```

◉ a5

○ a2

○ None of these

○ a1

○ a3

Question 23   1 / 1 pts

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

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

◉ int x = *p;

○ int y = a * b;

○ None of these use indirection.

○ z *= a;

○ int *p = &b;

Question 24   1 / 1 pts

What is the address of the first pixel in the last row of this image?

```cpp
Pixel *p;     // address of pixel data
int w, h;     // width and height of image
```

○ p + w + h

◉ p + w * (h - 1)

○ None of these are correct

○ p + w * h

○ p + w + (h - 1)

Incorrect **Question 25**   0 / 1 pts

Which expression returns the number of countries?

```cpp
string countries[] = {"Andorra", "Albania", . . . };
```

○ sizeof(countries) * sizeof(countries[0])

○ sizeof(countries)

○ len(countries)

○ sizeof(countries) / sizeof(string)

◉ None of these