Midterm 3 Study Guide

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

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 26	16 minutes	25 out of 25
LATEST	Attempt 26	16 minutes	25 out of 25
	Attempt 25	18 minutes	24 out of 25
	Attempt 24	15 minutes	25 out of 25
	Attempt 23	21 minutes	24 out of 25
	Attempt 22	24 minutes	24 out of 25
	Attempt 21	18 minutes	24 out of 25
	Attempt 20	30 minutes	19 out of 25
	Attempt 19	20 minutes	23 out of 25
	Attempt 18	30 minutes	21 out of 25
	Attempt 17	30 minutes	23.5 out of 25
	Attempt 16	30 minutes	24 out of 25
	Attempt 15	30 minutes	21 out of 25
	Attempt 14	30 minutes	23 out of 25
	Attempt 13	24 minutes	22 out of 25
	Attempt 12	16 minutes	24 out of 25
	Attempt 11	17 minutes	19 out of 25
	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 20 at 1:19pm

Question 1

1 / 1 pts

What happens with the following section of code?

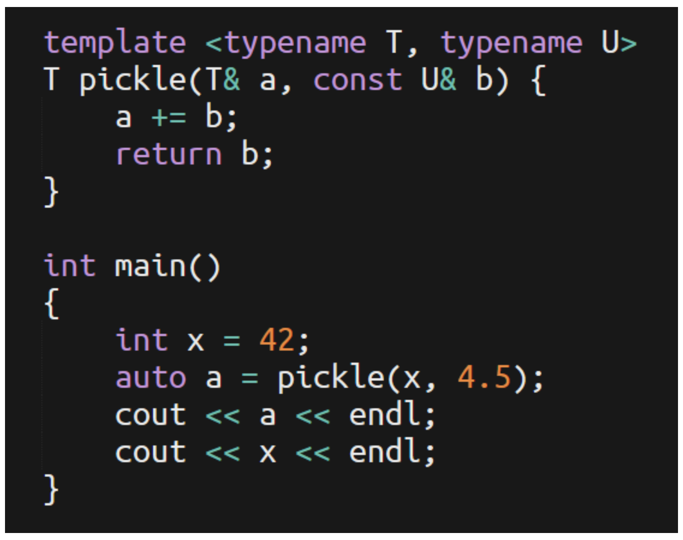
```
cout << "Enter 1, 2 or 3: ";
int n;
cin >> n;
#if 1
    cout << "You entered 1" << endl;
#elif 2
    cout << "You entered 2" << endl;
#elif 3
    cout << "You entered 3" << endl;
#else
    cout << "Invalid value" << endl;
#endif
```

- [✓ Compiles, but always print "You entered 1"]
- Compiles, but only prints "Invalid value"
- Compiles and prints the correct value entered by the user.
- Does not compile

Question 2

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;
}
```

- In main, x prints 46.5
- In main, a prints 4.5
- This code has a syntax error.
- [✓ In main, a prints 4]
- [✓ In main, x prints 46]

Question 3

1 / 1 pts

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

- [✓ True]
- False

Question 4

1 / 1 pts

The preprocessor operates on code **after** it has been compiled.

- True
- [✓ False]

Question 5

1 / 1 pts

Which line fails to work correctly?

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

- [✓ None of these]
- `print(2 + 2);`
- `print(string("goodbye"));`
- `print(3 + 2.2);`
- `print("hello");`

Question 6

1 / 1 pts

A function template may be declared in a header file but **must be** defined in an implementation file.

- True
- [✓ False]

Question 7

1 / 1 pts

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

- True
- [✓ False]

Question 8 1 / 1 pts

To use different versions of a function depending on the platform is called ***conditional compilation***.

- [✓ True]
- False

Question 9 1 / 1 pts

In the ***flag-controlled-pattern***, you use a `break` statement to exit the loop when the sentinel is found.

- True
- [✓ False]

Question 10 1 / 1 pts

When using the STL function `count_if`, the third argument is:

- `cbegin(v)`
- the value to count
- `cend(v)`
- [✓ a predicate function]
- None of these

Question 11 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.
- `void mystery(vector&);`
- [✓ `void mystery(vector<int>&);`]

Question 12 1 / 1 pts

What prints when this code runs?

```
enum class Coin
{
   PENNY = 1, NICKEL, DIME, QUARTER
};
cout << static_cast<int>(Coin::DIME) << endl;
```

- `2`
- [✓ `3`]
- `10`
- Does not compile; Missing semicolon at end of list of members.

Question 13 1 / 1 pts

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

- [✓ True]

- False

Question 14
1 / 1 pts

The `push_back` member function adds elements to the end of a `vector` expanding the `vector's` capacity if needed.

- [✓ True]
- False

Question 15
1 / 1 pts

The elements of a `vector` are allocated contiguously.

- [✓ True]
- False

Question 16
1 / 1 pts

Assume that you have an iterator named `iter` which refers to an element in the `vector` named `v`. Which moves the iterator so that it refers to the **next** element in the `vector`?

- [✓ `++iter;`]
- `iter.next();`
- None of these
- `*iter;`
- `iter++;`

Question 17
1 / 1 pts

The `push_back` member function adds elements to the end of a `vector`.

- [✓ True]
- False

Question 18
1 / 1 pts

The value for the variable ***b*** is stored:

```
int a = 1;
void f(int b)
{
    int c = 3;
    static int d = 4;
}
```

- [✓ on the stack]
- in the CPU machine registers
- in the static storage area
- The example does not provide enough information
- on the heap

Question 19
1 / 1 pts

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

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

- None of these are correct
- `p + w + (h - 1)`
- [✓ `p + w * (h - 1)`]
- `p + w * h`

○ p + w + h

Question 20

1 / 1 pts

The value for the variable ***d*** is stored:

```
int a = 1;
void f(int b)
{
    int c = 3;
    static int d = 4;
}
```

○ The example does not provide enough information

○ on the heap

[✓ in the static storage area]

○ in the CPU machine registers

○ on the stack

Question 21

1 / 1 pts

Assume that ***ppi*** correctly points to ***pi***. Which line prints the value stored inside ***pi***?

```
int main()
{
    double pi = 3.14159;
    double *ppi;
    // code goes here
    // code goes here
}
```

○ `cout << *pi;`

○ `cout << &ppi;`

○ `cout << &pi;`

○ `cout << ppi;`

[✓ None of these]

Question 22

1 / 1 pts

Array subscripts are not range checked

[✓ True]

○ False

Question 23

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++;`]

○ `p1++;`

○ `*p3 = 7;`

○ `*p1 = 3;`

○ `p2++;`

Question 24

1 / 1 pts

The allocated size of a built-in C++ array cannot be changed during runtime.

[✓ True]

○ False

Question 25 1 / 1 pts

Examine this version of the ***swap()*** function. How do you call it?

```
void swap(int * x, int & y)
{
    . . .
}
. . .
int a = 3, b = 7;
// What goes here ?
```

- ○ `swap(&a, &b);`
- [✓ `swap(&a, b);`]
- ○ `swap(a, b);`
- ○ `swap(a, &b);`
- ○ None of these