

Midterm 2 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 7	28 minutes	21 out of 25
LATEST	Attempt 7	28 minutes	21 out of 25
	Attempt 6	24 minutes	21 out of 25
	Attempt 5	20 minutes	17.83 out of 25
	Attempt 4	30 minutes	17.67 out of 25
	Attempt 3	22 minutes	16 out of 25
	Attempt 2	16 minutes	16.17 out of 25
	Attempt 1	30 minutes	18.33 out of 25

⚠️ Correct answers are hidden.

Submitted Jun 28 at 1:06am



Question 11 / 1 pts

In the classic *for* loop, which portion of code is analogous to an *if* statement?

- ☒ condition expression
- ☐ assignment statement
- ☐ first statement following the loop
- ☐ initialization statement
- ☐ None of these
- ☐ update expression

Incorrect

Question 20 / 1 pts

The highlighted section below illustrates. While more-characters:

```

Given: the variable str is a string (may be empty)
Create the counter variable, initialized to -1
If the variable str has any characters then
{
    Set counter to 0
    Create the variable current-character as a character
    Place the first character in str into current-character
    While more-characters and current-character not a period
    {
        Add one to (or increment) the counter variable
        Store the next character from str in current-character
    }
    If current-character is a period then
        Add one to the counter to account for the period.
    Else
        Set counter to -2
}
If counter is -1 the string was empty
Else if counter is -2 there was no period

```

- ☐ a necessary condition
- ☐ an intentional condition
- ☐ a postcondition
- ☐ a loop guard
- ☒ a boundary condition
- ☐ None of these

Incorrect

Question 3

0 / 1 pts

Below is the illustration from the loop building strategy. The **highlighted lines** represent. If current-character is a period then:

```

Given: the variable str is a string (may be empty)
Create the counter variable, initialized to -1
If the variable str has any characters then
{
    Set counter to 0
    Create the variable current-character as a character
    Place the first character in str into current-character
    While more-characters and current-character not a period
    {
        Add one to (or increment) the counter variable
        Store the next character from str in current-character
    }
    If current-character is a period then
        Add one to the counter to account for the period.
    Else
        Set counter to -2
}
If counter is -1 the string was empty
Else if counter is -2 there was no period

```

- ☐ loop postcondition
- ☐ bounds precondition
- ☐ advancing the loop
- ☐ loop bounds
- ☐ goal precondition

- ☒ goal operation

Question 4

1 / 1 pts

Loop bounds used when searching through input.

- ☒ sentinel bounds
- ☐ limit bounds
- ☐ None of these
- ☐ data bounds

Question 5

1 / 1 pts

Below is the illustration from the loop building strategy. The **highlighted lines** represent. Set counter to 0:

```
Given: the variable str is a string (may be empty)
Create the counter variable, initialized to -1
If the variable str has any characters then
{
    Set counter to 0
    Create the variable current-character as a character
    Place the first character in str into current-character
    While more-characters and current-character not a period
    {
        Add one to (or increment) the counter variable
        Store the next character from str in current-character
    }
    If current-character is a period then
        Add one to the counter to account for the period.
    Else
        Set counter to -2
}
If counter is -1 the string was empty
Else if counter is -2 there was no period
```

- ☐ loop bounds
- ☐ bounds precondition
- ☐ advancing the loop
- ☒ goal precondition
- ☐ goal operation
- ☐ loop postcondition

Question 6

1 / 1 pts

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

mechanics?

- ☒ loop bounds
- ☐ post condition
- ☒ advancing the loop
- ☐ goal precondition
- ☐ goal operation
- ☒ bounds precondition

Incorrect

Question 7

0 / 1 pts

What prints?

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

- ☐ Does not compile
- ☒ Hello
- ☐ Crashes when run
- ☐ Undefined behavior
- ☐ Hell

Incorrect

Question 8

0 / 1 pts

An ***unguarded*** loop is also known as a ***test-at-the-bottom*** loop.

- ☐ True
- ☒ False

Question 9

1 / 1 pts

Below is the illustration from the loop building strategy. The ***highlighted lines*** represent. Add one to (or increment) the counter variable:

```

Given: the variable str is a string (may be empty)
Create the counter variable, initialized to -1
If the variable str has any characters then
{
    Set counter to 0
    Create the variable current-character as a character
    Place the first character in str into current-character
    While more-characters and current-character not a period
    {
        Add one to (or increment) the counter variable
        Store the next character from str in current-character
    }
    If current-character is a period then
        Add one to the counter to account for the period.
    Else
        Set counter to -2
    }
}
If counter is -1 the string was empty
Else if counter is -2 there was no period

```

☐ advancing the loop

☐ loop postcondition

☐ loop bounds

☐ goal precondition

☒ goal operation

☐ bounds precondition

Question 10

1 / 1 pts

What prints here?

```

auto a = 3, b = 3;
cout << (a != b ? "panda": "tiger") << endl;

```

☐ panda

☐ Undefined behavior

☐ Does not compile

☐ Crashes when run

☒ tiger

Question 11

1 / 1 pts

What prints here?

```

auto a = 1;
switch (a)
{
    case 1: cout << "1";
    case 2: cout << "2";
}

```

```
}  
cout << endl;
```

☐ 2

☐ 1

☒ 12

☐ Does not compile

☐ Undefined behavior

Question 12

1 / 1 pts

Default arguments appear only in the function implementation.

☐ True

☒ False

Question 13

1 / 1 pts

Examine this code. Which is the best prototype?

```
int age;  
string name = read("Enter your name, age: ", age);
```

☒ string read(const string&, int&)

☐ string read(const string&, int)

☐ string read(const string, int&)

☐ string read(string, int);

☐ None of these

Question 14

1 / 1 pts

In a library, the ***implementation*** file:

☒ consists of function definitions

☐ consists of function calls

☐ consists of instructions that produce the executable

- ☐ consists of declarations or prototypes
- ☐ None of these

Question 15

1 / 1 pts

An incomplete, yet compilable, linkable and executable function is called a _____ ?

- ☐ None of these
- ☒ stub
- ☐ declaration
- ☐ prototype

Question 16

1 / 1 pts

Different functions that have the same name, but take different arguments, are said to be:

- ☒ overloaded
- ☐ overridden
- ☐ default
- ☐ covariant
- ☐ derived

Question 17

1 / 1 pts

Which of these documentation tags are used in a ***file comment***?

- ☐ @param
- ☐ @code
- ☒ @file
- ☒ @date

Question 18

1 / 1 pts

Examine the code below and match the statements following it.

```
int mystery3(int n) {
    if (n < 2) return 1;
    return n * mystery3(n - 1);
}
```

mystery3 has a stack overflow for some numbers.

False

mystery3 correctly implements its algorithm

True

if (n < 2) is a . . .

base case

mystery3 is efficient

True

mystery3 is an implementation of the Factorial algorithm. It completes for all inputs, but negative inputs produce the wrong output. It is efficient and it is not a wrapper. if (n < 2) is a base case.



Question 19

1 / 1 pts

The redirection pipe symbol is a pair of vertical bars (| |).

☐ True

☒ False

Question 20

1 / 1 pts

One remarkably simple formula for calculating the value of π is the so-called Madhava–Leibniz series: $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots$. Consider the recursive function below to calculate this formula:

```
double computePI(int number)
{
    if (number <= 1) { return 1.0;}
    int oddnum = 2 * number - 1;
    return computesign(number) * 1.0 / oddnum
        + computePI(number - 1);
}
```

In this recursive function, what is the role of the helper function computesign?

- ☒ it makes sure the sign (positive or negative) alternates as each term of the series is computed
- ☐ it is called just one time to set the sign of the final result
- ☐ it is the recursive call in the function
- ☐ it checks the sign of the number and returns true if it is positive and false if negative

Question 21

1 / 1 pts

Assume the user types "brown cow" when this code runs. What prints?

```
int n;  
if (cin >> n) cout << "X\n";  
else cout << "Y\n";
```

- ☐ x
- ☐ Runtime exception thrown
- ☒ y
- ☐ Does not compile

Question 22

1 / 1 pts

Stream parameters should always be passed to functions by ***const reference***.

- ☐ True
- ☒ False

Question 23

1 / 1 pts

Match the following code the the answers below.

```
template <typename T, typename U>
U 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;
}
```

Inside main, the variable a is type:

double



Inside main, the value printed for a is:

4.5



Inside main, the value printed for x is:

46



Question 24

1 / 1 pts

The file `grades.txt` contains lines of text that look like this:

```
Smith 94
Jones 75
. . .
```

Each line of text contains the student's name (a single word) and an integer score. What is the legal way of reading one student's information, given the following code?

```
string name;
int score;
ifstream in("grades.txt");
```

☐ `in << name << score;`

☐ None of these

☒ `in >> name >> score;`

☐ `getline(in, name); getline(in, score);`

☐ `getline(in, name); in >> score;`

Question 25**1 / 1 pts**

Infinite recursion can occur because

- ☒ the base case is missing one of the necessary termination conditions
- ☐ a second function is called from the recursive one
- ☐ the recursive function is called more than once
- ☐ the recursive case is invoked with simpler arguments

