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 11	29 minutes	25 out of 25
LATEST	Attempt 14	22 minutes	24 out of 25
	Attempt 13	25 minutes	24 out of 25
	Attempt 12	30 minutes	22 out of 25
	Attempt 11	29 minutes	25 out of 25
	Attempt 10	30 minutes	20.17 out of 25
	Attempt 9	29 minutes	20 out of 25
	Attempt 8	29 minutes	20.5 out of 25
	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 8:16pm

Question 1

1 / 1 pts

This loop uses asymmetric bounds.

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

- (•) True
- ( ) False

Question 2

1 / 1 pts

A ***guarded*** loop is also known as a ***test-at-the-top*** loop.

- (•) True
- ( ) False

Question 3

1 / 1 pts

Which of these is a ***flow-of-control*** statement?

- [ ] `x++;`
- [ ] `int x;`
- [x] `while (x < 3) …`

- [x] `for (auto e : s) …`
- [ ] `int y{15};`
- [x] `if (x < 3) … else …`

Question 4

1 / 1 pts

Below is the illustration from the loop building strategy. The ***highlighted lines*** represent.
Store the next character from str in current-character:

```
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
```

- ( ) goal operation
- (•) advancing the loop
- ( ) bounds precondition
- ( ) loop bounds
- ( ) loop postcondition
- ( ) goal precondition

Question 5

1 / 1 pts

In the classic *for* loop, which portion of code is not followed by a semicolon?

- ( ) condition expression
- (•) update expression
- ( ) initialization statement
- ( ) None of these

Question 6

1 / 1 pts

Below is the illustration from the loop building strategy. The ***highlighted lines*** represent.
While more-characters and current-character not a period:

```
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
- bounds precondition
- (●) loop bounds
- goal precondition
- goal operation
- loop postcondition

Question 7

1 / 1 pts

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

- True
- (●) False

Question 8

1 / 1 pts

Match each item with the correct statement below.

Keeps processing input until a particular value is found in input.	sentinel loop
Keeps processing until the output gets no closer to the answer.	limit loop
Repeats its actions a fixed number of times	definite loop
Keeps processing until the input device signals that it is finished.	data loop

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
```

- ○ bounds precondition
- ○ loop postcondition
- ○ goal precondition
- ○ advancing the loop
- ◉ goal operation
- ○ loop bounds

Question 10

1 / 1 pts

Header guards:

- ☑ go in every interface file
- ☐ includes the directive `#if`
- ☐ go in every client file
- ☐ start with the directive `#ifdef`
- ☑ includes the directive `#define`
- ☐ go in every implementation file
- ☑ end with the directive `#endif`
- ☑ start with the directive `#ifndef`

Question 11

1 / 1 pts

Given the ***overloaded*** functions prototypes and the variable definition below, which of the function calls will fail to compile?

```
int f(int&);
int f(const int&);
int f(int, int);
int a = 7;
```

- ☐ `f(3)`
- ☑ None of these fail to compile
- ☐ `f(2.0);`
- ☐ `f('a', 'b')`
- ☐ `f(a);`

Question 12

1 / 1 pts

In a `while` loop, (*`condition`*) is followed by a semicolon.

- ○ True
- ◉ False

Question 13

1 / 1 pts

Match each item with the correct statement below.

End a block of source code	@endcode
Required to document functions, global variables and constants.	@file
Your name	@author
When was it created?	@date

Question 14

1 / 1 pts

Which prototypes in the following header file contain errors?

```
#ifndef EXAMPLE_H
#define EXAMPLE_H
#include <string>

string f1(int a);
int f2(double);
void f3(std::string& s, int n);
double f4();

#endif
```

- ☐ *f2*
- ☑ *f1*
- ☐ None of these
- ☐ *f3*
- ☐ *f4*

Question 15

1 / 1 pts

What prints here?

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

- ◉ `43210`
- ○ Syntax error: `i` is not a Boolean expression

- 4321
- 54321
- Infinite loop

Question 16

1 / 1 pts

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

- None of these
- stub
- prototype
- declaration

Question 17

1 / 1 pts

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

- `@file`
- `@code`
- `@param`
- `@version`

Question 18

1 / 1 pts

Which line runs the `dwk` program and gets its input from a file named `y.data`?

- `./dwk >> y.data`
- `./dwk < y.data`
- `./dwk | y.data`
- `./dwk << y.data`
- `./dwk > y.data`
- `None of these`

Question 19

1 / 1 pts

The return value of the `getline()` function is an input stream object.

- True
- False

Question 20

1 / 1 pts

Complete the following code in the upper filter program.

```
char ch;
while (cin.get(ch))
    cout.put(_____________);
```

- ch + 32
- toupper(ch)
- toUpper(ch)
- ch - ('a' - 'A')

Question 21

1 / 1 pts

The cin object is an instance of the istream class.

- True
- False

Question 22

1 / 1 pts

Examine the code below:

```
int mystery1(int n, int a, int b) {
    if (n == 0) return a;
    if (n == 1) return b;
    return mystery1(n - 1, b, a + b);
}

int mystery2(int n) {
    return mystery1(n, 0, 1);
}
```

- [x] mystery1 is a recursive helper
- [ ] mystery1 is a recursive wrapper
- [x] The algorithm implemented is Fibonacci
- [x] if (n==1) is a base case

mystery2 is a recursive wrapper around the recursive helper mystery1. Together they implement the Fibonacci sequence in an efficient manner. mystery2 will not complete for any negative inputs.

Incorrect

Question 23

0 / 1 pts

When using the get() member function to read a character, leading whitespace is not skipped.

- True
- False

Question 24

1 / 1 pts

One remarkably simple formula for calculating the value of π is the so-called Madhava–Leibniz series: $\frac{\pi}{4} = 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\ldots$ 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 recursive base case?

- (●) When the parameter variable is less than or equal to one
- ( ) When the parameter variable is zero
- ( ) When the value that is returned from the function is zero
- ( ) When the parameter variable is greater than one

Question 25

1 / 1 pts

The operating system stream `stdin` is connected to your monitor by default.

- ( ) True
- (●) False