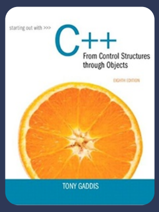
Study ⌄ ···

CS150 Practice Exam 2

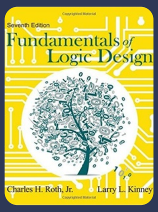
Share

11 studiers today ★ Leave the first rating

Textbook solutions for this set



Starting Out with C++ from Control Structures to Objects
8th Edition • ISBN: 9780133769395 (12 more)
Godfrey Muganda, Judy Walters, Tony Gaddis
1,294 solutions



Fundamentals of Logic Design
7th Edition • ISBN: 9781133628477 (2 more)
Jr. Charles H. Roth, Larry L Kinney
1,140 solutions

Search for a textbook or question >

Terms in this set (329)

End a block of source code Meaning of value returned from a function Required to document functions, global variables and constants Begin a block of source code Your name Information about the library When was it created? Name and meaning for a parameter	@endcode @return @file @code @author @version @date @param
Which of these documentation tags are used in a file comment?	@version @author @date @file
Which of these documentation tags are used in a function comment?	@return @param @endcode @code
What kind of error is this? ex1.cpp:7:10: error: expected ';' after expression a = 4 ^ ;	Syntax error (mistake in grammar)
What kind of error is this? ex1.cpp:6:5: error: use of undeclared identifier 'a' a = 4; ^	Compiler error (something is missing when compiling)

CS150 Practice Exam 2

ex1.cpp:6:12: error: no viable conversion from 'int' to 'string' string a = 15; ^ ~~	
What kind of error is this? ex1.cpp:7:9: warning: missing terminating '"' character a = "hello world'; ^ ex1.cpp:7:9: error: expected expression	Syntax error (mistake in grammar)

What is the output of the following?
 	
string s = "12345";
int i = 1;
while (i < 5)
{
cout << s.substr (i, 1);
i++;
}
	2345

What is the output of the following?
 	
string s = "abcde";
int i = 1;
while (i < 5)
{
cout << s.substr (i, 1);
i++;
}
	bcde

What is the output of the following?
 	
int i = 1;
while (i < 10)
{
cout << i << " ";
i = i + 2;
if (i == 5)
{
i = 9;
}
}
	1 3 9

What is the output of the following?
 	
int i = 1;
while (i <= 10)
{
cout << "Inside the while loop" << endl;
i = i * 11;
}
	"Inside the while loop" will be displayed only once.

What is the output of the following?
 	
int i = 1;
int sum = 0;
while (i <= 11)
{
sum = sum + i;
i++;
}
cout << "The value of sum is " << sum;
	The value of sum is 66

```
int i = 0;
while (i != 11)
{
cout << i << " ";
i = i + 2;
}
```

What is the output of the following?

```
int i = 1;
int sum = 0;
while (i <= 13)
{
sum = sum + i;
i = i + 3;
}
cout << "The value of sum is " << sum;
```

The value of sum is 35

How many times will this display "So far so good"?

```
int i = 0;
while (i != 15)
{
cout << "So far so good" << endl;
i++;
}
```

15 times

What is the output of the following?

```
int i = 1;
while (i < 20)
{
cout << i << " ";
i = i + 2;
if (i == 15)
{
i = 19;
}
}
```

1 3 5 7 9 11 13 19

What is the output of the following?

```
int i = 0, j = 0;
while (i < 125)
{
i = i + 2;
j++;
}
cout << j << endl;
```

63

Header files must explicitly qualify each name from the standard library with std::

True

Header files may use the statement using namespace std;

False

An undefined error message is a linker error.

True

An undefined error message is a compiler error

False

An undeclared error message is a linker error	False
Implementation files may use the statement using namespace std; Implementation files must explicitly qualify each name from the standard library with std::	True False
Parameter names are optional in the function prototype Parameter names are optional in the function definition	True False
A tool named Doxygen is often used to generate HTML user docs from C++ code. If a prototype in a header file has a parameter that is a library type, the header file must #include the appropriate library header.	True True

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
	f1

Which prototypes in the following header file contain errors?
 	
#ifndef EXAMPLE_H
#define EXAMPLE_H
 	
string f1(int a);
int f2(double);
void f3(std::string& s, int n);
double f4();
 	
#endif
	f1
	f3

Which prototypes in the following header file contain errors?
 	
#ifndef EXAMPLE_H
#define EXAMPLE_H
#include <string>
 	
std::string f1(int a);
int f2(double);
void f3(std::string& s, int n);
double f4();
 	
#endif
	None of these

EXE=digit-tester
OBJS=client.o digits.o
$(EXE): $(OBJS)
$(CXX) $(CXXFLAGS) $(OBJS) -o $(EXE)	

Which of these are targets? EXE=digit-tester OBJS=client.o digits.o $(EXE): $(OBJS) $(CXX) $(CXXFLAGS) $(OBJS) -o $(EXE)	$(EXE) digit-tester

How many lines of output are printed?
 	
int i = 0;
while (i != 9)
{
cout << "Loop Execution" << endl;
i++;
}
	9

What is the output of the following?
 	
int i = 0;
while (i != 9)
{
cout << i << " ";
i = i + 2;
}
	0 2 4 6 8 10 12 14 .... (infinite loop)

What is the output of the following?
 	
int i = 1;
while (i != 9)
{
cout << i << " ";
i++;
if (i = 9)
{
cout << "End";
}
}
	1 End

How many lines of output are printed?
 	
int count = 0;
while (count != 9)
{
cout << "Monster Mash" << endl;
if ((count % 2) == 0)
{
count++;
}
else
{
count--;
}
}
	Infinite

bool token = false;
while (token)
{
cout << "Hello World!" << endl;
}	

What is the output of the following?
 	
bool token1 = true;
while (token1)
{
for (int i = 0; i < 5; i++)
{
cout << "Hello there" << endl;
}
token1 = false;
}
	"Hello there" will be displayed 5 times.

What is the output of the following?
 	
bool val1 = true;
bool val2 = false;
while (val1)
{
if (val1)
{
cout << "Hello" << endl;
}
val1 = val2;
}
	"Hello" will be displayed only once.

Which line in the function "skeleton" below contains an error?
 	
#include "digits.h" // 1.
int firstDigit(int n); // 2.
{ // 3.
return 0; // 4.
} // 5.
	// 2.

Which line in the function "skeleton" below contains an error?
 	
#include "digits.h" // 1.
int firstDigit(int n) // 2.
{ // 3.
return 0; // 4.
}
	None of these

Which line in the function "skeleton" below contains an error?
 	
#include "borgia.h" // 1.
void primoTiara(int n) // 2.
{ // 3.
return 0; // 4.
} // 5.
	// 4.

What kind of error is this? ex1.cpp:7: undefined reference to `f()'	Linker error (something is missing when linking)
What kind of error is this? ~/workspace/ $ ./ex1 The Patriots won the 2018 Super Bowl	None of these

CS150 Practice Exam 2

Study

terminate called after throwing an instance of 'std::out_of_range'	
What kind of error is this? Segmentation fault	Operating system signal or trap

In a library, the implementation file:	consists of function definitions
In a library, the interface file:	consists of declarations or prototypes
In a library, the client or test program:	consists of function calls
In a library, the makefile:	consists of instructions that produce the executable
In a client file you should compare your function's value to the _______?	expected value
In a client file, the value returned from calling your function is the_______?	actual value
Loops that do some processing and then compare the results against a boundary condition are called _________?	limit loops
An incomplete, yet compilable, linkable and executable function is called a ______ ?	stub
Which of these program organization schemes does not work?	Call your functions and define them afterwards.
Which of these may go into a header file?	function prototypes constant definitions

When you call a function, the compiler must know:	the number of arguments to pass the name of the function the type of each argument the kind of value returned if any
Header guards:	end with the directive #endif includes the directive #define go in every interface file start with the directive #ifndef

Object file	digits.o
Library file	libdigits.a
Interface file	digits.h
Project file	makefile
Client file	digit tester.cpp
Implementation file	digits.cpp

What prints?

```
void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }

int main()
{
auto n = 3.5;
fn(1, 2.5, n);
}
```

A

What prints?

```
void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }

int main()
{
fn(2.5, 1.5, 2.5);
}
```

C

What prints?

```
void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }

int main()
{
fn(1, 2, 3.5);
}
```

C

What prints?

```
void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }

int main()
{
fn(2.5, 1.5, 7);
}
```

D

void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }
 	
int main()
{
fn(1, 2, 3, 4);
}	

What prints?
 	
void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }
 	
int main()
{
auto n = 3.5;
fn(1, 2, n);
}
	Syntax error: ambiguous

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

What prints here?
 	
auto a = 4, b = 3;
cout << (a == b ? "panda": a % 2 ? "stork": "tiger") << endl;
	tiger

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

What prints here?
 	
auto a = 3, b = 3;
cout << (a != b ? "panda": a % 2 ? "stork": "tiger") << endl;
	stork

What prints here?
 	
auto a = 3, b = 3;
cout << a == b ? "panda" : "tiger" << endl;
	Does not compile

Function overloading allows you to write several different functions that have the same name.	True
Function overloading lets you call a single function in several different ways.	False
Overloaded functions have the same name but different parameter types.	True
Overloaded functions have the same name but different parameter names.	False
In a while loop, (condition) is followed by a semicolon.	False
A while loop is a hasty or unguarded loop.	False

auto a = 1;
switch (a)
{
case 1: cout << "1"; break;
case 2: cout << "2"; break;
default: cout << "3";
}
cout << endl;	

What prints here?
 	
auto a = 2;
switch (a)
{
case 1: cout << "1"; break;
case 2: cout << "2"; break;
default: cout << "3";
}
cout << endl;
	2

What prints here?
 	
auto a = '1';
switch (a)
{
case 1: cout << "1"; break;
case 2: cout << "2"; break;
default: cout << "3";
}
cout << endl;
	3

What prints here?
 	
auto a = 1;
switch (a)
{
case 1: cout << "1";
case 2: cout << "2";
}
cout << endl;
	12

What prints here?
 	
auto a = 1;
switch (a)
{
case 1: cout << "1";
case 2: cout << "2";
case 3:
}
cout << endl;
	Does not compile

What prints here?
 	
double a = 1;
switch (a)
{
case 1: cout << "1";
case 2: cout << "2";
}
cout << endl;
	Undefined behavior

CS150 Practice Exam 2

auto a = 'A';
switch (a)
{
case 64: cout << "?";
case 65: cout << "A";
case 66: cout << "B";
}
cout << endl;	

The compiler determines which overloaded function to call by looking at the number, types and order of the arguments passed to the function.	True
Default arguments let you call a single function in several different ways. Default arguments allow you to write several different functions that have the same name.	True False
Default arguments may only be used with value parameters. Default arguments may only be used with reference parameters. Default arguments may be used with both value and reference parameters.	True False False
Default arguments appear only in the function prototype. Default arguments appear only in the function implementation.	True False
Fatal error messages should be printed to cerr. Fatal error messages should be printed to cout.	True False
Calling break() terminates a program immediately and passes an error code back to the operating system.	False
The compiler determines which overloaded function to call by looking at the type of value the function returns.	False
If str = "hello", then str.size() > -1.	False
Calling exit() terminates a program immediately and passes an error code back to the operating system.	True
A parameter with a default argument cannot appear before a parameter without a default argument.	True
A do-while loop is also called a hasty loop.	True
In a do-while loop, (condition) is followed by a semicolon.	True

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);
}
	string&

void f(. . . str);
int main()
{
f("hello");
}	

To allow f() to change the argument passed here, the parameter str should be declared as:
 	
void f(. . . str);
int main()
{
f("hello");
}
	It is not possible for f() to change the argument passed here.

A function where an argument is converted to match a parameter When more than one match is found for the proffered arguments. A function where each argument is the same type as the corresponding parameter. A group of functions with the same name. A group of functions that have the same name and the correct number of parameters. When no match is found for the proffered arguments	best match ambiguity exact matches candidate set viable set empty set

Examine the following variables and function calls
Match each item with the correct statement below.
int able = 3;
int baker = f1(able);
cout << able << baker << endl; // 64
 	
int charlie;
f2("hello", charlie);
cout << charlie << endl; // Hello Carl
	Returned value --> baker
	
	Output argument (parameter) --> Charlie
	
	Input argument (parameter) --> Hello
	
	Input/output argument (parameter) --> able

Which of these are not ways that functions may be overloaded?	different function name different return type different parameter names
Different functions that have the same name, but take different arguments, are said to be:	overloaded
You can call a single function in several different ways by giving the function _________:	default arguments

Given the overloaded functions prototypes and the variable definition below, which of the function calls will fail to compile?
 	
int f(int&);
int f(int);
int f(int, int);
int a = 7;
	f(a);

compile?
 	
int f(int&);
int f(const int&);
int f(int, int);
int a = 7;	

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;
	2

Assume that the input is 5 5 4 3 5. What will print?
 	
int i = 1;
int n;
do
{
cin >> n;
i++;
}
while (n % 2);
cout << i << endl;
	4

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

Examine this code. Which is the best prototype? int age; string name = read("Enter your name, age: ", age);	string read(const string&, int&)

What prints?
 	
string str = "Hello";
for (int i = str.size() - 1; i >= 0; i--)
cout << str.at(i);
	olleH

What prints?
 	
string str = "Hello";
for (size_t i = str.size() - 1; i >= 0; i--)
cout << str.at(i);
	Crashes when run

What prints?
 	
string str = "Hello";
for (auto i = 0, len = str.size(); i < len; i++)
cout << str.at(i);
	Does not compile

CS150 Practice Exam 2

int main()
{
string str{"To be or not to be."};
cout << "Most common letter is "
<< mostCommon(str) << endl;
}	

Which of these prototypes is the best one to use in this circumstance?
 	
int main()
{
string str{"TO BE OR NOT TO BE"};
properCase(str);
cout << str << endl;
}
	void properCase(string&);

Examine this code. Which is the best prototype?
 	
string s = "dog";
cout << upper(s) << endl; // DOG
cout << s << endl; // dog
	string upper(const string&)

Examine this code. Which is the best prototype?
 	
string s = "dog";
upper(s);
cout << s << endl; // DOG
	string upper(const string&)

Arguments passed to a function that has a non-constant reference parameter must be:	lvalues
A named constant, which can only be initialized once, is known as a ______________.	non-modifiable lvalue
Arguments passed to a function that has a constant reference parameter must be:	either lvalues or rvalues are fine
The pattern of parameter types and order is called the function's:	signature

What prints here?
 	
int i = 5;
while (--i) cout << i;
cout << endl;
	4321

What prints here?
 	
int i = 5;
while (i--) cout << i;
cout << endl;
	43210

What prints here?
 	
int i = 5;
while (i) cout << --i;
cout << endl;
	43210

What prints here?
 	
int i = 5;
while (i) cout << i--;
cout << endl;
	54321

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

The input stream member function for reading a character at a time is named:	get()
Assume you have a char variable named ch. How do you read one character from input?	cin.get(ch);
The expression cin.get(ch) does which of these?	reads the next character in input and stores it in ch returns a reference to cin that can be tested
Assume you have a char variable named ch. How do you "unread" a character already read?	cin.putback(ch);
Assume you have a char variable named ch. How do you write one character to output?	cout.put(ch);

Complete the following code in the echo filter program.
 	
char ch;
while (cin.get(ch))
__________;
	cout.put(ch)

Complete the following code in the lower filter program.
 	
char ch;
while (cin.get(ch))
cout.put(________);
	tolower(ch)

Complete the following code in the upper filter program.
 	
char ch;
while (cin.get(ch))
cout.put(________);
	toupper(ch)

Complete the following code in the echo filter program.
 	
char ch;
while (_________)
cout.put(ch);
	cin.get(ch)

Assume the user types "brown cow" when this code runs. What type is ch2?
 	
char ch1;
auto ch2 = cin.get(ch1);
	istream&

Assume the user types "brown cow" when this code runs. What prints?
 	
int n;
if (cin >> n) cout << "X\n";
else cout << "Y\n";
	Y

Assume the user types "brown cow" when this code runs. What is stored in ch2?
 	
char ch1;
auto ch2 = cin.get(ch1);
	cin

char c;
cout.put(cin.get(c));	

Assume the user types "brown cow" when this code runs. What prints?
 	
char c;
cout << cin.get(c) << endl;
	Does not compile

When using cin >> ch; to read a character, leading whitespace is skipped.	True
When using cin >> ch; to read a character, leading whitespace is not skipped.	False
Calling cout.put(65) prints the character 'A' on output	True
Calling cout.put(65) prints the number 65 on output	False
Calling cout.put(65) is illegal. Your code will not compile.	False
Calling cout.put(65.35) is illegal. Your code will not compile	False
When using the get() member function to read a character, leading whitespace is not skipped	True
When using the get() member function to read a character, leading whitespace is skipped.	False
A process filter does something to the characters it encounters	True
A process filter learns something about the stream by examining characters	False
The expression cin.get(ch) returns a reference to the input stream	True
The expression cin.get(ch) returns the next character from input	False
A state filter learns something about the stream by examining characters	True
A state filter does something to the characters it encounters	False
Counting the number of words in input by counting word transitions is an example of a state filter	True
Counting the number of words in input by counting word transitions is an example of a process filter.	False
You can test if an I/O operation succeeded by explicitly calling the stream's fail() member function	True
To test if an I/O operation succeeded you must explicitly call the stream's fail() member function	False
Calling cout.put(c) converts its argument, c, to a character.	True
Calling cout.put("A") is illegal. Your code will not compile.	True

CS150 Practice Exam 2

When using the get() member function, a stream will fail only if there are no characters left in the input stream.	True
Programs that process streams of characters are called text ________.	filters
Which of these are not process filters?	compress input by turning off echo when reading blank spaces print one sentence per line counting word transitions
Which of these are not state filters?	translating data from one form to another search for a particular value in a stream copy a file
Assume you have a char variable named ch. How do you look ahead before reading a character?	cin.peek();

Assume you have a char variable named ch. How do you look ahead before reading a character?
 	
2 Q U E S T I O N S
	cin.get(ch);
	cin.unget(ch);
	cin.putback(ch);
	cin.seek(ch);
	cin.peek(ch);
	
	-- > None of these

Which line runs the dwk program and gets its input from a file named y.data?	./dwk < y.data
Which line runs the prt program and stores its output in a new file named x.data?	./prt > x.data
Which line runs the dmm program and adds its output to a file named x.data?	./dmm >> x.data
Which line runs the dd program and sends its errors to file named z.data?	./dd 2> z.data
Which line runs a.out getting its input from in.txt and appending its output to the file out.txt?	./a.out > in.txt >> out.txt
Which line runs a.out getting its input from in.txt and sending its output to the new file out.txt?	./a.out > out.txt < in.txt
Append output to a file named z Discard both output and errors Write output to a new file named z Read the input from the file named z Write errors to a new file named z Send the output to the input of the program named z	X rm x > /dev/null/2>&1 X cat < z cat x 2>z date | z
Which line runs the dom program and sends both output and errors to file named v.data?	./dom > v.data 2>&1

Returns the last character read to the input stream	unget()
Examines, but does not read the next character in an input stream	peek()
Replaces the last character read with any character	putback()
Called implicitly when an input statement is used as a test condition.	fail()
A predicate function	isalpha()
Converts its value argument to a character and sends it to output.	put()
Which line runs a.out getting its input from in.txt and sending its output to the file out.txt, and its errors to the file err.txt?	./a.out < in.txt > out.txt 2> err.txt
Indefinite limit loop that reduces its input	while (n!=0) {n/=2;}
Indefinite limit loop that uses successive approximations	while(abs(g1-g2) >= EPSILON) {...}
Counter-controlled symmetric loop for producing a sequence of data	for (int i = 12; i <= 19; i ++) {...}
Indefinite data loop that uses raw input	while(cin.get(ch)) {...}
Counter-controlled asymmetric loop for processing characters	for (size_t i = 0, len = s.size(); i < len; i++) {...}
Iterator loop that may change its container	for(auto&e : col) {...}
Iterator loop that cannot change its container	for(auto e: col) {...}
Counter-controlled loop for processing substrings	for(size_t i=4, slen =4; len = s.size(); i <len; i++) {...}
Indefinite data loop that uses formatted input	while(cin >> n)
A loop that reads data until some special value is found is called a:	sentinel loop
Which of these is not a technique for implementing a sentinel loop?	the counter-controlled pattern
What Java and other OO languages call a subclass, C++ calls a ________.	derived class
Stream arguments to a function should:	be as general as possible (istream and ostream)
Stream arguments to a function should always be passed:	by reference

The file temp.txt contains "Orange Coast College". What prints?
 	
ifstream in("temp.txt");
char c;
while (in.get(c))
{
if (isupper(c))
cout << toupper(c);
}
	OCC

Create an input file stream object named in.	ifstream in;
Which line opens the file in.txt for reading?	ifstream in("in.txt");

CS150 Practice Exam 2

Study

Create an input file stream object named in and open the text file "tuba.txt", using a single statement.	ifstream in("tuba.txt");
Create an output file stream object named out.	ofstream out;
Which line opens the file out.txt for writing?	ofstream out; out.open("out.txt");
Create an output file stream object named out and open the text file "expenses.dat", using a single statement.	ofstream out("expenses.dat");
Use the output stream object named out to create the text file on disk named "totals.txt".	out.open("totals.txt");
Establish an association between the input stream object named in, and the text file on disk named "pets.txt".	in.open("pets.txt");

Which line reads a single word from the istream named in into the string variable word?
 	
word = in.next();
in.get(word);
getline(in, word);
in << word;
None of these
	None of these

The file temp.txt contains "If I saw an Aardvark, I would scream!". What prints?
 	
ifstream in("temp.txt");
char c;
int i = 0;
while (in.get(c))
{
if (tolower(c) == 'a') i++;
}
cout << i << endl;
	6

The return value of the getline() function is an input stream object The return value of the getline() function is a string object.	True False
When writing a function with stream parameters, always use the most general type of stream that meets the specification When writing a function with stream parameters, always use the most specific type of stream that meets the specification	True False
The cout object is an instance of the ostream class. The cout object is an instance of the ofstream class	True False
A loop that reads data until the input stream signals that it is done is called a data loop A loop that reads data until the input stream signals that it is done is called a sentinel loop	True False

CS150 Practice Exam 2

Study

In the primed loop pattern, you use Boolean flag to signal when the sentinel is found	False
In the primed loop pattern, you use a break statement to exit the loop when the sentinel is found	False
The getline() function is a non-member function in the string library	True
The getline() function is a member function in the string class	False
The getline() function is a member function in the istream class.	False
To use a disk file as a data stream source or sink, use the <fstream> header	True
To use a disk file as a data stream source or sink, use the <ifstream> header	False
To use a disk file as a data stream source or sink, use the <ofstream> header	False
Unformatted I/O means that you read and write data character-by-character	True
Unformatted I/O means that you read and write data line-by-line	False
Formatted I/O means that you read and write data token-by-token	True
Formatted I/O means that you read and write data line-by-line	False
The C++ term for what is called a superclass in other languages is base class	True
The C++ term for what is called a superclass in other languages is derived class	False
The cin object is an instance of the istream class	True
The cin object is an instance of the ifstream class	False
Stream parameters should always be passed to functions by reference	True
Stream parameters should always be passed to functions by const reference	False
In the flag-controlled-pattern, you use Boolean variable to signal when the sentinel is found	True
In the flag-controlled-pattern, you use a break statement to exit the loop when the sentinel is found.	False
In the flag-controlled-pattern, you read data before the loop and at the end of the loop	False

In the loop-and-a-half, you use Boolean variable to signal when the sentinel is found	False
In the loop-and-a-half pattern, you read data before the loop and at the end of the loop.	False
If an input stream's file is missing when you try to open it, its fail() member function returns true	True
If an input stream's file is missing when you try to open it, its fail() member function returns false	False
If an output stream's file is missing when you try to open it, its fail() member function returns false.	True
To use strings as a data stream source or sink, use the <sstream> header	True
To use strings as a data stream source or sink, use the <stringstream> header	False
The C++ term for what is called a subclass in other languages is derived class	True
The C++ term for what is called a subclass in other languages is base class	False
A loop that reads data until some special value is found is called a sentinel loop.	True
A loop that reads data until some special value is found is called a data loop.	False
To read a line of text, you include the header file <string>	True
A token is a "chunk of meaningful data".	True
In the C++ stream hierarchy, the base class of the ifstream class is:	istream
In the C++ stream hierarchy, the base class of the ofstream class is:	ostream
In the C++ stream hierarchy, the base class of the ostream class is:	ios
In the C++ stream hierarchy, base class of the istream class is:	ios
In the C++ stream hierarchy, the base class of the stringstream class is:	iostream
In the C++ stream hierarchy, the base class of the fstream class is:	iostream

CS150 Practice Exam 2

Study

Connect a disk file to an input or output stream	fstream
Use the predefined stream objects cin and cout	iostream
Determine the category of a character	cctype
Modify the way that memory is converted to characters on input or output	iomanip

Which fragment completes this code segment?
 	
string fmt(double n, int decimals)
{
ostringstream out;
out << fixed << setprecision(decimals);
out << n;
return _________;
}
	out.str()

After writing data to an ostringstream object named os, you can retrieve the string it contains by using:	os.str()

What does this code do?
 	
ifstream in("temp.txt");
char x;
int i{0};
while (in.get(x)) i++;
cout << i << endl;
	Counts the number of characters in the file

What does this code do?
 	
ifstream in("temp.txt");
string x;
int i{0};
while (getline(in, x)) i++;
cout << i << endl;
	Counts the number of lines in the file

What does this code do?
 	
ifstream in("temp.txt");
string x;
int i{0};
while (in >> x) i++;
cout << i << endl;
	Counts the number of words in the file

Which of the following loop patterns are used here?
 	
size_t pos = 0;
char ch;
in.get(ch);
while (ch != 'Q')
{
pos++;
in.get(ch);
}
	primed loop
	sentinel loop

Which of the following loop patterns are used here?
 	
int upper = 0;
char ch;
while (in.get(ch))
{
if (ch >= 'A' && ch <= 'Z')
upper++;
}
	inline test
	data loop

int n;
in >> n;
while (abs(n))
{
out << n % 4 << endl;
n /= 4;
}	

Which of the following loop patterns are used here?
 	
auto len = str.size();
while (len) out << str.at(--len);
	counter-controlled loop

Which of the following loop patterns are used here?
 	
string s{"hello CS 150"};
for (auto e : s)
{
if (toupper(e))
out.put('x');
}
	iterator or range loop

Which of the following loop patterns are used here?
 	
string s{"hello CS 150"};
for (auto e : s)
{
if (toupper(e)) break;
}
	iterator or range loop
	
	loop-and-a-half

Which of the following loop patterns are used here?
 	
string s{"Hello CS 150"};
while (s.size())
{
if (s.at(0) == 'C') break;
s = s.substr(1);
}
cout << s << endl;
	counter-controlled loop
	
	loop-and-a-half
	
	sentinel loop

After opening the input stream in, which of these cannot be used to see if the file was successfully opened?	if (in.opened()) {/ **opened ok** /}

This loop:
 	
char c;
while (in.get(c))
{
cout << c << endl;
}
	illustrates raw character I/O

This loop:
 	
char c;
while (c = in.get())
{
cout << c << endl;
}
	illustrates line-based stream processing

This loop:
 	
string str;
while (getline(in, str))
{
cout << str << endl;
}
	illustrates line-based stream processing

string str;
while (in >> str)
{
cout << str << endl;
}	

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;

The file expenses.txt contains the line: Hotel, 3 nights. $ 1,750.25. What prints?
 	
ifstream in("expenses.txt");
char c;
while (in.get(c))
{
if (isdigit(c)) {
in.unget();
double n;
in >> n;
cout << n << 'x';
}
}
	3x1x750.25x

The file expenses.txt contains the line: Hotel, 3 nights. $ 1,750.25. What prints?
 	
ifstream in("expenses.txt");
char c;
while (in.get(c))
{
if (isdigit(c)) {
in.unget();
int n;
in >> n;
cout << n << 'x';
}
}
	3x1x750x25x

Assume that the file scores.txt does not exist. What happens?
 	
ofstream out("scores.txt");
out << "Peter" << " " << 20 << endl;
out << "John" << " " << 50 << endl;
	Creates a new file, scores.txt and writes two lines of text.

Which line represents the necessary bounds in this 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;	2

CS150 Practice Exam 2

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;	
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;	5
What header file to you need to include to use the standard C++ error-handling classes?	<stdexcept>
The logic_error and runtime_error classes are defined in the header file ___.	stdexcept

What prints?
 	
string s("hello");
try {
auto x = s.at(s.size()); 🔅
cout << "one" << endl;
}
catch (const string& e) { cout << "two\n"; }
catch (exception& e) { cout << "three\n"; }
catch (...) { cout << "four\n"; }
	three

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

What prints?
 	
string s("hello");
try {
if (s.size() > 2) throw s.size(); 🔅
if (islower(s.back())) throw s.back(); 🔅
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"; }
 	
➢ I F (s.size() > 2) && throw s.size() && throw s.back()
	five

CS150 Practice Exam 2

Study

```
string s("hello");
try {
if (s.size() > 5) throw s.size(); 🔆
if (isupper(s.back())) throw s.back(); 🔆
if (s == "hello") throw string("hello");
s.at(s.size()) = 'x';
cout << "one\n";
}
catch (const string& e) { cout << "two\n"; }
catch (exception& e) { cout << "three\n"; }
catch (...) { cout << "four\n"; }
```

➢ I F (s.size() > 5) && throw s.size() && throw s.back()

What prints?

```
string s("hello");
try {
if (s.size() > 2) 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"; }
```

➢ I F (s.size() > 2) && throw 42; && throw "goodbye";

two

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"; }
```

➢ I F (s.size() > 20) && throw 42; && (islower(s.back())) throw "goodbye";

five

CS150 Practice Exam 2

Study

```
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"; }
```

➢ I F (s.size() > 2) && throw 42; && (isupper(s.back())) throw "goodbye";

What is correct for # 1?

```
int main()
{
//1
{
string s = "hello";
cout << s.at(5) << endl;
}

// 2
// 3
( e)
{
cout << e. () << endl;
// 4
}

}
```

try

What is correct for # 2?

```
int main()
{
//1
{
string s = "hello";
cout << s.at(5) << endl;
}

// 2
// 3
( e)
{
cout << e. () << endl;
// 4
}

}
```

catch

int main()
{
//1
{
string s = "hello";
cout << s.at(5) << endl;
}
 	
// 2
// 3
(e)
{
cout << e. () << endl;
// 4
}
 	
}	

What is correct for # 4?
int main()
{
//1
{
string s = "hello";
cout << s.at(5) << endl;
}
 	
// 2
// 3
(e)
{
cout << e. () << endl;
// 4
}
 	
}
	what

The C++11 standard library provides the function stoi() to convert a string to an integer. Which library is it found in?	string
What preprocessor directive is not used when you wish to create blocks of code that are only compiled under certain circumstances?	#define #ifdef #ifndef #if --> All of these may be used
Code that may cause an error should be placed in a _________ block and code that handles the error should be inside a _______ block?	try, catch
The class ___ is the base of the classes designed to handle exceptions	exception
A(n) ___ is an occurrence of an undesirable situation that can be detected during program execution	exception
What statement is used to signal other parts for your program that a particular error has occurred?	throw
The class ___ is designed to deal with illegal arguments used in a function call.	invalid_argument
What is the purpose of the throw statement?	It is used to pass control to an error handler when an error situation is detected.
The try block is followed by one or more ___ blocks.	catch

The function ___ returns a string containing an appropriate message.	what
A catch block can have, at most, ___ catch block parameter(s).	one
What happens when this code fragment runs in C++ 11? cout << sqrt(-2) << endl;	sqrt() returns a not-a-number error value
Variables tested with the #if preprocessor directive are created using #define	True
Without try and catch, the throw statement terminates the running program	True
A try block is a block of code where runtime or logical errors may occur	True
A catch(...) will catch any kind of thrown exception	True
Functions with generic parameters are known as function templates.	True
A completion code is a special return value that means "the function failed to execute correctly."	True
Calling a function like to_string(3.5) is known as implicit instantiation	True
To use different versions of a function depending on the platform is called conditional compilation.	True
Building your code with more than one copy of a function leads to a clash of symbols.	True
A template function may be defined in a header file.	True
The predefined constant _cpluplus indicates which version of the C++ standard is being used	True
One of the main problems with the completion code strategy of error handling is that callers can ignore the return value without encountering any warnings	True
Calling a function like to_string<int>(3.5) is known as implicit instantiation.	False
The line: cin >> n; throws a runtime exception if n is an int and it tries to read the input "one".	False
The preprocessor operates on code after it has been compiled.	False
The directives #if defined(symbol) and #ifdef symbol mean, essentially, the same thing The directives #if defined(symbol) and #ifndef symbol mean, essentially, the same thing.	True False

A catch block may only handle objects from classes derived from exception or logic_error	False
A catch block specifies the type of exception it can catch and immediately terminates the program	False
A catch block is a block of code where runtime or logical errors may occur	False
You can report a logical error encountered in your code by using the throw keyword	True
You can report a syntax error encountered in your code by using the throw keyword	False
Functions with generic parameters may use the keyword class or the keyword typename for their type parameters	True
Functions with generic parameters may use the keyword class or the keyword struct for their type parameters	False
The #if preprocessor directive can compare integers	True
The #if preprocessor directive may compare double literals but not variables	False
The standard library version of sqrt(-2) returns the not-a-number error code	True
The standard library version of sqrt(-2) throws a runtime exception because there is no possible answer	False
You compiler or contains constants that can be used to identify the platform you are compiling on	True
A specialized error handling block of code, is called a catch block	True
A specialized error handling block of code, is called a try block	False
The standard library version of stoi("UB-40") throws a runtime exception because there is no viable conversion	True
The standard library version of stoi("UB-40") returns the not-a-number error code.	False
The order of the catch blocks does not affect the program.	False
If no exception is thrown in a try block, all catch blocks associated with that try block are ignored.	True
When you throw an exception, control immediately jumps out of the current try block.	True
The preprocessor operates on code before it has been compiled.	False
The statement #if abs(-3) > 2 is legal.	False
A template function may be declared in a header file but must be defined in an implementation file.	False

When you throw an exception, control immediately returns from the current function	False
The line: ifstream in("x"); throws a runtime exception if a file x cannot be found	False
What happens when this code fragment runs? cout << stoi("12") << endl;	stoi() returns 12
What happens when this code fragment runs in C++ 11? cout << stoi("one") << endl;	It throws a runtime exception
Which of the following statements throws a valid exception in C++?	throw 2;
Suppose you have written a program that inputs data from a file. If the input file does not exist when the program executes, then you should choose which option?	Terminate the program.

What happens when this code fragment runs?
 	
istringstream in("12.5");
int n;
in >> n;
	n is set to 12

What happens when this code fragment runs?
 	
istringstream in("12");
int n;
in >> n;
	n is set to 12

What happens when this code fragment runs?
 	
istringstream in(".5");
int n;
in >> n;
	It sets an error state in in.

What happens when this code fragment runs in C++ 11?
 	
istringstream in("one");
int n;
in >> n;
	It sets an error state in in.

To deal with logical errors in a program, such as string subscript out of range or an invalid argument to a function call, several classes are derived from the class ___.	logic_error

Which line fails to work correctly?
 	
template <typename T>
void print(const T& item)
{
cout << item << endl;
}
	ANSWER --> None of these
	print(2 + 2);
	print(string("goodbye"));
	print(3 + 2.2);
	print("hello");

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

Which call below produces 5?
 	
template <typename T>
void addem(T a, T b)
{
cout << a << " + " << b << "->"
<< (a + b) << endl;
}
	addem<int>(3, 2.5);

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;
}
	ANSWER --> None of these
	addem(1.5, 2);
	addem(s1, s2);
	addem(3, 4)
	addem(4.5, 5.5);

What happens when this code fragment compiles and runs? #define N #ifdef N cout << "Hello"; #else cout << "Goodbye"; #endif	prints "Hello"
What happens when this code fragment compiles and runs? #define N #ifndef N cout << "Hello"; #else cout << "Goodbye"; #endif	prints "Goodbye"

What term describes this block of code?
 	
#if _APPLE_
istringstream in(" .75");
int n = 3;
in >> n;
#endif
	conditional compilation

Complete the code fragment below, which is designed to throw an illegal_length exception if string variable accountNumber has more than seven characters.
 	
if (accountNumber.size() > 7)
{
____________________;
}
	throw illegal_length("Account number exceeds maximum length");

Examine the following code (which is legal). What is the correct prototype for an aggregate output operator? struct Time { int hours{0}, minutes{0}, seconds{0}; };	ostream& operator<<(ostream& out, const Time& m);

struct Money { int dollars{0}, cents{0}; } m1, m2;	
Examine the following code (which is legal). Which statement is illegal? struct Money { int dollars{0}, cents{0}; } m1, m2;	cout << m1 << endl;
Examine the following code (which is legal). Which statement is legal? struct Money { int dollars{0}, cents{0}; } m1, m2;	m1 = m2;
Examine the following code (which is legal). Which statement is correct? struct Rectangle { int length, width; };	Rectangle r;
The following is legal. Which is the correct way to access a data member in the Rectangle variable named r? struct Rectangle { int length, width; };	r.length
The structure and variable definitions are fine. Which statements are legal? struct Rectangle { int length, width; } big, small;	if (big.length == small.width) . . .
The following is legal. Which changes the length data member inside the variable big? struct Rectangle { int length, width; } big, little;	big.length = 10;

Examine the following code (which is legal). What changes are necessary to allow the statement if (m1 == m2) ... to compile?
 	
struct Money { int dollars{0}, cents{0}; } m1, m2;
 	
bool equals(const Money& lhs, const Money& rhs)
{
return lhs.cents == rhs.cents &&
lhs.dollars == rhs.dollars;
	The name of equals() must be changed to operator==

Examine the following code (which is legal). What changes are necessary to allow the statement if (m1 != m2) ... to compile?
 	
struct Money { int dollars{0}, cents{0}; } m1, m2;
 	
bool equals(const Money& lhs, const Money& rhs)
{
return lhs.cents == rhs.cents &&
lhs.dollars == rhs.dollars;
}
	You must write a function named operator!=

Examine the following definition. What is the syntax error?
 	
struct Employee
{
long empID;
std::string lastName;
double salary;
}
	missing a semicolon after the structure definition

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

Examine the following definition. Employee is the ________.
 	
struct Employee
{
long empID;
std::string lastName;
double salary;
};
	structure tag

Given the following structure and variable definitions, which data members are uninitialized?
 	
struct Employee
{
long empID{0};
std::string lastName;
double salary{0};
int age = 0;
};
 	
Employee bob;
	None of them (compiles)

Given the following structure and variable definitions, which data members are uninitialized?
 	
struct Employee
{
long empID;
std::string lastName;
double salary;
int age;
};
 	
Employee bob;
	salary
	age
	empID

Given the following structure and variable definitions, which data members are initialized?
 	
struct Employee
{
long empID;
std::string lastName;
double salary;
int age;
};
 	
Employee bob;
	lastName

Given the following structure and variable definitions, which data members are initialized?
 	
struct Employee
{
long empID;
std::string lastName;
double salary;
int age;
};
 	
Employee bob{};
	salary
	age
	lastName
	empID

struct Employee
{
long empID;
std::string lastName;
double salary;
int age;
};
 	
Employee bob{777, "Zimmerman"};	

Given the following structure and variable definitions, which data members are default initialized?
 	
struct Employee
{
long empID;
std::string lastName;
double salary;
int age;
};
 	
Employee bob{777, "Zimmerman", 5000000.0, 76};
	None of these

Given the following structure and variable definitions which statements are legal?
 	
struct Money
{
int dollars{0};
int cents{1};
};
Money payment;
	cout << payment.dollars;
	payment.cents = 5;

Given the following structure and variable definitions which statements are illegal?
 	
struct Money
{
int dollars{0};
int cents{1};
};
Money payment;
	payment{1} = 5;
	cout << Money.dollars;
	Money{1} = Money{0};

The structure and variable definitions are fine. Which statements are legal?
 	
struct R { int a, b; } a, b;
struct Q { int a, b; } c, d;
	c = d;

Y O U D O N O T N E E D T O R E V I E W F O R T R U E / F A L S E 🚨	Y O U D O N O T N E E D T O R E V I E W F O R T R U E / F A L S E 🚨

CS150 Practice Exam 2

Structures are heterogeneous data types.

The built-in primitive data types such as int, char and double are scalar data types.

User-defined scalar types are created with the enum class keywords in C++.

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

The standard library types such as string and vector are structured data types.

You may create a structure variable as part of a structure definition.

The following is an anonymous structure.
struct {int hours, seconds; } MIDNIGHT{0, 0};

Structure variables should be passed to functions by reference.

When passing a structure variable to a function, use non-const reference if the intent is to modify the actual argument.

The following code is legal.
struct {int hours, seconds; } MIDNIGHT{0, 0};

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

A structure member may be a variable of a different structure type.

In C++, objects have value semantics; object variables contain the data members.

Structures data members may each have a different type.

C++ has two ways to represent records, the class and the struct.

This is the correct syntax for a C++ scoped enumeration.
enum class WEEKEND {SATURDAY, SUNDAY};

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

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

In Computer Science, a collection of variables that have distinct names and types is called a record.

This is the correct syntax for a C++ plain enumeration.
enum WEEKEND {SATURDAY, SUNDAY};

CS150 Practice Exam 2

Study

User-defined types that combine multiple values into a single type are called scalar types

It is legal to include the same struct definition multiple times, as long as the definitions are exactly the same.

In C++, objects have reference semantics; object variables refer to, but do not contain the data members.

A structure definition creates a new variable.

In C++, a collection of variables that have distinct names and types is called a record.

In C++, a collection of variables that have distinct names and types is called a structure.

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

This is the correct syntax for a C++ scoped enumeration.
enum WEEKEND {SATURDAY, SUNDAY};

Structure variables should be passed to functions by value.

User-defined scalar types are created with the struct or class keywords in C++.

Structures are homogenous data types.

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

Structures data members must all be of the same type.

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

The built-in primitive data types such as int, char and double are structured data types.

When passing a structure variable to a function, use const reference if the intent is to modify the actual argument.

The standard library types such as string and vector are scalar data types.

The following code is illegal.
struct {int hours, seconds; } MIDNIGHT{0, 0};