CH 09 Q U I Z, Mine, CH 07 Q U I Z, CH 08 Q U I Z, CH 05 Q U I Z, CH 06 Q U I Z, Chapter 19 C++ Study Guide

Share

10 studiers recently ★ Leave the first rating

Terms in this set (401)

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");
Which line opens the file input.txt for reading?	ifstream in("input.txt");
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

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

is found In the loop-and-a-half, you use Boolean variable to signal when the sentinel is found In the loop-and-a-half pattern, you read data before the loop and at the end of the loop.	False False
If an input stream's file is missing when you try to open it, its fail() member function returns true If an input stream's file is missing when you try to open it, its fail() member function returns false If an output stream's file is missing when you try to open it, its fail() member function returns false.	True False True

To use strings as a data stream source or sink, use the <sstream> header To use strings as a data stream source or sink, use the <stringstream> header	True False
The C++ term for what is called a subclass in other languages is derived class The C++ term for what is called a subclass in other languages is base class	True False
A loop that reads data until some special value is found is called a sentinel loop. A loop that reads data until some special value is found is called a data loop.	True 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
Read and write characters to memory using streams Connect a disk file to an input or output stream Use the predefined stream objects cin and cout Determine the category of a character Modify the way that memory is converted to characters on input or output	sstream fstream iostream cctype 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()

```
ifstream in("temp.txt");
char x;
int i{0};
while (in.get(x)) i++;
cout << i << endl;
```

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

Which of the following loop patterns are used here?

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

limit loop

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

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

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

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

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
An unguarded loop is also known as a test-at-the-top loop.	False
What information is produced?	goal precondition
Can my loop reach its bounds?	necessary bounds
How is the data processed?	loop operations or actions
What makes this loop quit?	loop bounds
Set counter to 0	goal precondition
In the classic for loop, which portion of code is not followed by a semicolon?	update expression
In the classic for loop, which portion of code is analogous to an if statement?	condition expression
In a guarded loop, the loop actions may never be executed.	True
In the classic for loop, loop control variables going from 0 to less-than n are said to employ:	asymmetric bounds
Using the loop-building strategy from the lessons, which of these are part of the loop mechanics?	bounds precondition loop bounds advancing the loop

Which of these are guarded loops?	while
	for

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

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

Overloaded functions have the same name but different parameter names.	False
Default arguments may only be used with value parameters.	True
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 here?
auto a = 3, b = 3;
cout << (a != b ? "panda": a % 2 ? "stork": "tiger") << endl;
	stork

What is the value of r("hello")?
string r(const string& s)
{
if (s.size() > 1) {
string t = s[0] == s[1] ? "" : "*";
return t + s[0] + r(s.substr(1));
}
return s;
}
	"***h**el*lo"

The input redirection symbol, << asks the operating system to open a file and pass its contents to your program as standard input.	False

What is the value of r(12777)?
int r(int n)
{
if (0 == n) return 0;
int x = n % 10 == 7; // 0 or 1
return x + r(n / 10);
}
	3

This command: cat < nofile 2> /dev/null will print an error message on the screen if nofile does not exist.	False

What is the value of r(126)?
int r(int n)
{
if (n >= 10) return n % 10 + r(n / 10);
return n;
}
	9

In 1735 Leonard Euler proved a remarkable result, which was the solution to the Basel Problem, first posed in 1644 by Pietro Mengoli. This result gave a simple expression for . The formula states that is equal to the limit, as n goes to infinity, of the series . Which statement below is the correct base case for a recursive implementation that approximates this infinite series?	if (number <= 1) { return 1.0;}
Which line represents the intentional 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;	4
A loop that reads data until some special value is found is called a:	sentinel loop
In a guarded loop, the loop actions are always executed at least once.	False

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

Header guards:	includes the directive #define end with the directive #endif go in every interface file start with the directive #ifndef
Default arguments appear only in the function prototype.	True
Object file	digits.o
Interface file	digits.h
Client file	digit tester.cpp
Implementation file	digits.cpp
Header files must explicitly qualify each name from the standard library with std::	True

Which of these prototypes is the best one to use in this circumstance?
int main()
{
string str{"To be or not to be."};
cout << "Most common letter is "
<< mostCommon(str) << endl;
}
	char mostCommon(const string&);

Which of these may go into a header file?	function prototypes constant definitions
The getline() function is a non-member function in the string library.	True

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

Which line runs the dom program and sends both output and errors to file named v.data?	./dom > v.data 2>&1
The Unix filter to use for searching through text to find a particular word is called grep.	True
Stream arguments to a function should always be passed:	by reference
Pipes redirect the output of one program to be the input to another program.	True
Loop bounds used when searching through input.	sentinel bounds
In the classic for loop, which portion is used to create the loop control variable?	initialization statement
A guarded loop is also known as a test-at-the-bottom loop.	False

current-character not a period	an intentional condition
In an unguarded loop, the loop actions are always executed at least once.	True

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)

Default arguments may only be used with reference parameters.	False

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.

Function overloading lets you call a single function in several different ways.	False
Meaning of value returned from a function	@return
Begin a block of source code	@code
Information about the library	@version
Name and meaning for a parameter	@param
In a library, the implementation file:	consists of function definitions

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.

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

Unformatted I/O means that you read and write data line-by-line.	False

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

Which line opens the file input.txt for reading?	ifstream in("input.txt");
Create the variable current-character as a character Place the first character in str into current-character	bound precondition
A guarded loop is also known as a test-at-the-top loop.	True

Which of these is a flow-of-control statement?	for (auto e : s) ...
	while (x < 3) ...
	if (x < 3) ... else ...

Which are the two major categories of loops?	definite loops indefinite loops
If current-character is a period then Add one to the counter to account for the period. Else Set counter to -2	loop postcondition

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

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

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

How many lines of output are printed?
int i = 0;
int j = 0;
while (i < 25)
{
i = i + 2;
j++;
}
cout << j << endl;
	13

File containing the declarations or prototypes	interface
Program which uses the functions in a library.	client
File containing the function definitions	implementation
File which contains instructions for building your program	makefile
Counting the number of words in input by counting word transitions is an example of a process filter.	False
When using cin >> ch; to read a character, leading whitespace is not skipped.	False
cat < f.txt > f.txt makes a copy of f.txt.	False

What does this function do?
int mystery(int n, int m)
{
if (n == 0) return m;
return m * 10 + mystery(n / 10) + n % 10;
}
	Computes the reverse of the input n

What is the value of mystery(12)?
int mystery(int n)
{
if (!n) return 0;
return 2 + mystery(n-1);
}
	24

What Java and other OO languages call a superclass, C++ calls a ________.	base class
What changes about this function if lines 4 and 5 are swapped? 1. void myfun(const string& word) 2. { 3. if (word.size() == 0) { return; } 4. myfun(word.substr(1)); 5. cout << word[0]; 6. }	reverses the order in which the characters of the string are printed

What is the value of r(74757677)?
int r(int n)
{
if (n) return (n % 10 == 7) + r(n / 10);
return 0;
}
	5

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);
}	

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

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, 4);
}
	Syntax error: no candidates

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

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

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

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

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

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&

To allow f() to accept the argument passed here, the parameter str should be declared as:
 	
void f(. . . str);
int main()
{
f("hello");
}
	const string&

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	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);

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;
	None of these fail to compile

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&)

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

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

Which of these prototypes is the best one to use in this circumstance?
 	
int main()
{
string str{"To be or not to be."};
cout << "Most common letter is "
<< mostCommon(str) << endl;
}
	char mostCommon(const string&);

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

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

The input stream member function for reading a character at a time is named:	get()

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

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

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. When using cin >> ch; to read a character, leading whitespace is not skipped.	True False
Calling cout.put(65) prints the character 'A' on output Calling cout.put(65) prints the number 65 on output Calling cout.put(65) is illegal. Your code will not compile. Calling cout.put(65.35) is illegal. Your code will not compile	True False False False
When using the get() member function to read a character, leading whitespace is not skipped When using the get() member function to read a character, leading whitespace is skipped.	True False
A process filter does something to the characters it encounters A process filter learns something about the stream by examining characters	True False

The expression cin.get(ch) returns the next character from input	False
A state filter learns something about the stream by examining characters A state filter does something to the characters it encounters	True False
Counting the number of words in input by counting word transitions is an example of a state filter Counting the number of words in input by counting word transitions is an example of a process filter.	True False
You can test if an I/O operation succeeded by explicitly calling the stream's fail() member function To test if an I/O operation succeeded you must explicitly call the stream's fail() member function	True 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
When a stream is converted to a Boolean condition, its fail() member function is implicitly called	True
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 I z
Which line runs the dom program and sends both output and errors to file named v.data?	./dom > v.data 2>&1

Has a single char& parameter
Returns the last character read to the input stream
Examines, but does not read the next character in an input stream
Replaces the last character read with any character
Called implicitly when an input statement is used as a test condition.
A predicate function
Converts its value argument to a character and sends it to output.
	unget()
	peek()
	putback()
	fail()
	isalpha()
	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
Indefinite limit loop that uses successive approximations
Counter-controlled symmetric loop for producing a sequence of data
Indefinite data loop that uses raw input
Counter-controlled asymmetric loop for processing characters
Iterator loop that may change its container
Iterator loop that cannot change its container
Counter-controlled loop for processing substrings
Indefinite data loop that uses formatted input
	while (n!=0) {n/=2;}
	while(abs(g1-g2) >= EPSILON) {...}
	for (int i = 12; i <= 19; i ++) {...}
	while(cin.get(ch)) {...}
	for (size_t i = 0, len = s.size(); i < len; i++) {...}
	for(auto&e : col) {...}
	for(auto e: col) {...}
	for(size_t i=4, slen =4; len = s.size(); i <len; i++) {...}
	while(cin >> n)

[1] What must I change in the test to go to the next iteration? [2] What information is produced? [3] What must I do to enter the loop? [4] Can my loop reach its bounds? [5] Has my loop reached its goal? [6] How is the data processed? [7] Can my loop be entered at all? [8] What makes this loop quit?	[1] advance the loop [2] goal precondition [3] bounds precondition [4] necessary bounds [5] loop postcondition [6] loop operations and actions [7] loop guards [8] loop bounds
[1] May not repeat its actions at all [2] Keeps processing input until a particular value is found in input. [3] Repeats its actions at least once [4] Keeps processing until the output gets no closer to the answer. [5] Test for the occurrence of a particular event [6] Repeats its actions a fixed number of times [7] Conditions under which a loop will repeat its actions [8] Keeps processing until the input device signals that it is finished.	[1] guarded loop [2] sentinel loop [3] unguarded loop [4] limit loop [5] indefinite loop [6] definite loop [7] loop bounds [8] data loop
[1] Actions that occur after the loop is complete [2] Actions occuring inside the loop's body [3] Actions that occur before the loop is encountered [4] A test that determines if the loop should be entered	[1] postcondition [2] operation [3] precondition [4] bounds

Which of these is a flow-of-control statement?	for (auto e : s) ...
	if (x < 3) ... else ...
	while (x < 3) ...

Which of these are guarded loops?	for
	while

Which of these are unguarded loops?	do-while
Which are the two major categories of loops?	definite indefinite
Which of these are indefinite loops?	sentinel bounds limit bounds data bounds
Using the loop-building strategy from Chapter 5, which of these are part of the loop mechanics?	loop bounds bounds precondition advancing the loop

[How many characters are in a sentence? Count the characters in a string until a period is encountered. If the string contains any characters, then it will contain a period. Count the period as well.]	
Look at the problem statement below. The ______ of the loop is that a period was encountered. [How many characters are in a sentence? Count the characters in a string until a period is encountered. If the string contains any characters, then it will contain a period. Count the period as well.]	bounds
Look at the problem statement below. The ______ of the loop is read a character and increment a counter. [How many characters are in a sentence? Count the characters in a string until a period is encountered. If the string contains any characters, then it will contain a period. Count the period as well.]	plan
Loop bounds used when searching through input.	sentinel bounds
Loop bounds often used in scientific and mathematical applications.	limit bounds
In the classic for loop, loop control variables going from 0 to less-than n are said to employ:	asymmetic bounds
Loop bounds used when reading files or processing network data.	data bounds

How many times is this loop entered? (That is, how many times is i printed?)
 	
for (int i = 1; i < 10; i++)
cout << i;
cout << endl;
	9

How many times is this loop entered? (That is, how many times is i printed?)
 	
for (int i = 1; i <= 10; i++)
cout << i;
cout << endl;
	10

How many times is this loop entered? (That is, how many times is i printed?)
 	
for (int i = 0; i < 10; i++)
cout << i;
cout << endl;
	10

How many times is this loop entered? (That is, how many times is i printed?)
 	
for (int i = 0; i <= 10; i++)
cout << i;
cout << endl;
	11

In the classic for loop, which portion of code is not followed by a semicolon?	update expression
In the classic for loop, which portion of code is executed after the last statement in the loop body?	update expression
In the classic for loop, which portion of code is analogous to an if statement?	condition expression
In the classic for loop, which portion is used to create the loop control variable?	initialization statement

```
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
```

Below is the illustration from the loop building strategy in Chapter 5. The highlighted lines represents:

```
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 precondition

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	

The highlighted selection below illustrates:
 	
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

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	

The highlighted selection below illustrates:
 	
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
	an intentional condition

Below is the illustration from the loop building strategy in Chapter 5. The highlighted lines represents:
 	
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

```
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
```

Below is the illustration from the loop building strategy in Chapter 5. The highlighted lines represents:

```
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

In a guarded loop, the loop actions may never be executed	True
In a guarded loop, the loop actions are always executed at least once.	False

---

In an unguarded loop, the loop actions are always executed at least once.	True
In an unguarded loop, the loop actions may never be executed.	False

---

A guarded loop is also known as a test-at-the-top loop	True
A guarded loop is also known as a test-at-the-bottom loop.	False

---

An unguarded loop is also known as a test-at-the-bottom loop.	True
An unguarded loop is also known as a test-at-the-top loop.	False

---

Loops are used to implement iteration in C++.	True
Loops are used to implement selection in C++.	False

for (int i = 1; i <= 10; i++)
cout << i;
cout << endl;
 	
This idiomatic pattern is used to count from one value to another.
 	
for (int i = 1; i < 10; i++)
cout << i;
cout << endl;
	False

This loop uses asymmetric bounds.
 	
for (int i = 0; i < 10; i++)
cout << i;
cout << endl;
	True

This loop uses asymmetric bounds.
 	
for (int i = 1; i < 10; i++)
cout << i;
cout << endl;
	True

This loop uses asymmetric bounds.
 	
for (int i = 1; i <= 10; i++)
cout << i;
cout << endl;
	False

End a block of source code	@endcode
Meaning of value returned from a function	@return
Required to document functions, global variables and constants	@file
Begin a block of source code	@code
Your name	@author
Information about the library	@version
When was it created?	@date
Name and meaning for a parameter	@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)
What kind of error is this? ex1.cpp:6:12: error: no viable conversion from 'int' to 'string' string a = 15; ^ ~~	Type error (wrong initialization or assignment)
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)

string s = "12345";
int i = 1;
while (i < 5)
{
cout << s.substr (i, 1);
i++;
}	

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

What is the output of the following?
 	
int i = 0;
while (i != 11)
{
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;
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

int i = 1;
while (i < 20)
{
cout << i << " ";
i = i + 2;
if (i == 15)
{
i = 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:: Header files may use the statement using namespace std;	True False
An undefined error message is a linker error. An undefined error message is a compiler error	True False
An undeclared error message is a run-time error An undeclared error message is a linker error	False 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

`#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`	

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

What is the output of the following?
 	
bool token = false;
while (token)
{
cout << "Hello World!" << endl;
}
	No output

bool token1 = true;
while (token1)
{
for (int i = 0; i < 5; i++)
{
cout << "Hello there" << endl;
}
token1 = false;
}	

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
What kind of error is this? terminate called after throwing an instance of 'std::out_of_range'	Runtime error (throws exception when running)
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 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
Executable Object file Library file Interface file Project file Client file Implementation file	digit-tester digits.o libdigits.a digits.h makefile digit tester.cpp digits.cpp

[2301] Given the function below, what does cout << mystery(3) print?
 	
int mystery(int n)
{
if (n < 2) return 1;
return n * mystery(n - 1);
}
6
120
2
24
	6

[2302] If you write mystery(10), how many times is the function called?
 	
int mystery(int n)
{
if (n <= 2) return 1;
return n * mystery(n - 1);
}
120
10
6
9
	9

[2303] What does this function do?
 	
int mystery(int n)
{
if (n == 1) return 1;
return n * mystery(n-1);
}
 	
Computes the reverse of the input n
Computes the Gauss series (sum) of 1..n
Computes the Factorial number n
Computes the Fibonacci number n
Produces a stack overflow
	Computes the Factorial number n

[2304] What does this function do?
 	
int mystery(int n)
{
if (n < 2) return 1;
return mystery(n-1) + mystery(n-2);
}
Computes the Gauss series (sum) of 1..n
Computes the Factorial number n
Computes the Fibonacci number n
Computes the reverse of the input n
Produces a stack overflow
	Computes the Fibonacci number n

int mystery(int n)
{
if (n == 1) return 1;
return n * mystery(n+1);
}
Computes the Gauss series of n
Computes the Fibonacci number n
Produces a stack overflow
Computes the Factorial number n
Computes the reverse of the input n	

[2306] What does this function do?
 	
int mystery(int n)
{
if (n == 1) return 1;
return n + mystery(n-1);
}
Computes the Factorial number n
Computes the reverse of the input n
Computes the Fibonacci number n
Produces a stack overflow
Computes the Gauss series (sum) of 1..n
	Computes the Gauss series (sum) of 1..n

[2307] What does this function do?
 	
int mystery(int n, int m)
{
if (n == 0) return m;
return m * 10 + mystery(n / 10) + n % 10;
}
 	
Produces a stack overflow
Computes the reverse of the input n
Computes the Factorial number n
Computes the Gauss series (sum) of 1..n
Computes the Fibonacci number n
	Computes the reverse of the input n

[2308] What is the value of mystery(12)?
 	
int mystery(int n)
{
if (!n) return 0;
return 2 + mystery(n-1);
}
 	
18
24
36
12
	24

[2309] What is the value of r(6)?
 	
int r(int n)
{
if (n > 0) return n + r(n - 1);
return n;
}
 	
15
6
10
24
21
	21

[2310] What is the value of mystery(5)?
 	
int mystery(int n)
{
if (n > 0) return 3 - n % 2 + mystery(n-1);
return 0;
}
7
12
5
10
15
	12

int r(int n)
{
if (n >= 10) return n % 10 + r(n / 10);
return n;
}
 	
3
6
13
10
9	

[2312] What is the value of r(12777)?
 	
int r(int n)
{
if (0 == n) return 0;
int x = n % 10 == 7; // 0 or 1
return x + r(n / 10);
}
5
Does not compile
2
3
Stack overflow
	3

[2313] What is the value of r(74757677)?
 	
int r(int n)
{
if (n) return (n % 10 == 7) + r(n / 10);
return 0;
}
 	
3
5
Does not compile
8
Stack overflow
	5

[2314] What is the value of r(74757677)?
 	
int r(int n)
{
if (n) return (n % 10 != 7) + r(n / 10);
return 0;
}
5
3
Does not compile
8
Stack overflow
	3

[2315] What is the value of r(8818)?
 	
int r(int n)
{
if (!n) return 0;
return (n % 10 == 8) + (n % 100 == 88) + r(n / 10);
}
Stack overflow
4
Does not compile
3
1
	4

[2316] What is the value of r(81238)?
 	
int r(int n)
{
if (!n) return 0;
return (n % 10 == 8) + (n % 100 == 88) + r(n / 10);
}
Does not compile
2
Stack overflow
5
3
	2

```
int r(int n)
{
if (!n) return 0;
return (n % 10 == 8) + (n % 100 == 88) + r(n / 10);
}
```
4
1
5
6
Stack overflow

[2318] What is the value of r(3, 3)?

```
int r(int n, int m)
{
if (m) return n * r(n, m - 1);
return 1;
}
```
12
27
Stack overflow
9
3

27

[2319] What is the value of r("xxhixx")?

```
int r(const string& s)
{
if (s.size())
return (s.at(0) == 'x') + r(s.substr(1));
return 0;
}
```

4
2
3
6
Stack overflow

4

[2321] What is the value of r("xxhixx")?

```
string r(const string& s)
{
if (s.empty()) return "";
if (s.at(0) == 'x') return 'y' + r(s.substr(1));
return s.at(0) + r(s.substr(1));
}
```

xxyyxx
yyhiyy
xyxyhixyxy
yxyxhixyyx
Stack overflow

yyhiyy

[2322] What is the value of r("xhixhix")?

```
string r(const string& s)
{
if (s.size()) {
auto c = s.at(0);
auto t = c == 'x' ? 'y' : c;
return t + r(s.substr(1));
}
return 0;
}
```
Stack overflow
yyyyyyy
xyyxyyx
yhiyhiy
xyhixyhixy

yhiyhiy

[2323] What is the value of r("axxbxx")?

```
string r(const string& s)
{
auto front = s.substr(0, 1);
if (front.empty()) return "";
return (front == "x" ? "" : front) + r(s.substr(1));
}
```
"a b "
"xxxx"
"ax bx "
"ab"
Stack overflow

"ab"

```
string r(const string& s)
{
auto front = s.substr(0, 1);
if (front.empty()) return "";
return (front == "x" ? front : "") + r(s.substr(1));
}
```

"ax bx "
"a b "
Stack overflow
"xxxx"
"ab"

[2325] Assume you have the array: int a[] = {1, 11, 3, 11, 11};.
What is the value of r(a, 0, 5)?

```
int r(const int a[], size_t i, size_t max)
{
if (i < max) return (a[i] == 11) + r(a, i + 1);
return 0;
}
```

3
5
Stack overflow
1
0

3

[2326] What is the value of r("hello")?

```
string r(const string& s)
{
if (s.size() < 2) return s;
return s.substr(0, 1) + "*" + r(s.substr(1));
}
```

"hell*o"
"hello*"
"hello"
Stack overflow
"hello"

"hello"

[2327] What is the value of r("hello")?

```
string r(const string& s)
{
if (s.size() > 1) {
string t = s[0] == s[1] ? "*" : "";
return s[0] + t + r(s.substr(1));
}
return s;
}
```

"hello"
Stack overflow
"hell*o"
"hello"
"hel*lo"

"hel*lo"

[2328] What is the value of r("hello")?

```
string r(const string& s)
{
if (s.size() > 1) {
string t = s[0] == s[1] ? "" : "*";
return s[0] + t + r(s.substr(1));
}
return s;
}
```

"hell*o"
"hel*lo"
"hello"
Stack overflow
"hello"

"h*e*ll*o"

```
string r(const string& s)
{
if (s.size() > 1) {
string t = s[0] == s[1] ? "" : "*";
return t + s[0] + r(s.substr(1));
}
return s;
}
```

"h**e**ll**o**"
Stack overflow
"h**e**ll*o"
"hel*lo"

"*h*el*lo"

[2330] Which of the following statements is correct about a recursive function?
A recursive function must never call another function.
A recursive function calls itself.
A recursive function must be simple.
A recursive function must call another function.

A recursive function calls itself.

[2331] What does this function do?

```
void myfun(string word)
{
if (word.length() == 0) return;
myfun(word.substr(1, word.length()));
cout << word[0];
}
```

Prints the length of the string word
Prints the string word both forward and reverse
Prints the string word in reverse
Prints the string word

Prints the string word in reverse

[2332] What changes about this function if lines 4 and 5 are swapped?

```
1. void myfun(string word)
2. {
3. if (word.length() == 0) { return; }
4. myfun(word.substr(1, word.length()));
5. cout << word[0];
6. }
```

prints the characters of the string in both forward and reverse order
creates infinite recursion
nothing
reverses the order in which the characters of the string are printed

reverses the order in which the characters of the string are printed

---

[2333] Which of the following is true about using recursion?

Recursion always helps you create a more efficient solution than other techniques.

A recursion eventually exhausts all available memory, causing the program to terminate

A recursive computation solves a problem by calling itself with simpler input.

None of the listed options.

A recursive computation solves a problem by calling itself with simpler input.

[2334] How can you ensure that a recursive function terminates?

Call the recursive function with simpler inputs.
Use more than one return statement.
Provide a special case for the simplest inputs.
Provide a special case for the most complex inputs.

Provide a special case for the simplest inputs

[2335] Which of the following is a key requirement to ensure that recursion is successful?

Every recursive call must simplify the computation in some way

A recursive solution should not be implemented to a problem that can be solved iteratively

There should be special cases to handle the most complex computations directly

A recursive function should not call itself except for the simplest inputs

Every recursive call must simplify the computation in some way.

int r(int n)
{
if (n < 2) { return 1; }
return n * r(n - 1);
}
 	
24
2
120
6	

[2337] Which statement ensures that r() terminates for all values of n?
 	
int mr(int n)
{
// code goes here
return r(n - 1) + n * n;
}
if (n == 1) { return 1; }
if (n == 0) { return 0; }
 	
if (n == 0) { return 0; }
 	
if (n < 1) { return 1; }
 	
if (n == 1) { return 1; }
	if (n < 1) { return 1; }

[2338] Infinite recursion can lead to an error known as stack overflow heap exhaustion heap fragmentation memory exception	stack overflow
[2339] Infinite recursion can occur because the base case is missing one of the necessary termination conditions the recursive function is called more than once the recursive case is invoked with simpler arguments a second function is called from the recursive one	the base case is missing one of the necessary termination conditions

[2340] Two quantities a and b are said to be in the golden ratio if mc040-1.jpg is equal to mc040-2.jpg. Assuming a and b are line segments, the golden section is a line segment divided according to the golden ratio: The total length (a + b) is to the longer segment a as a is to the shorter segment b. One way to calculate the golden ratio is through the continued square root (also called an infinite surd): golden ratio = mc040-3.jpg. In a recursive implementation of this function, what should be the base case for the recursion?
if (number <= 1) { return pow(number, 2.0);}
if (number <= 1) { return sqrt(number);}
if (number <= 1) { return 0.0;}
if (number <= 1) { return 1.0;}
	if (number <= 1) { return 1.0;}

[2341] Two quantities a and b are said to be in the golden ratio if mc041-1.jpg is equal to mc041-2.jpg. Assuming a and b are line segments, the golden section is a line segment divided according to the golden ratio: The total length (a + b) is to the longer segment a as a is to the shorter segment b. One way to calculate the golden ratio is through the continued square root (also called an infinite surd): golden ratio
 	
If the function double golden (int) is a recursive implementation of this function, what should be the recursive call in that function?
return sqrt (1.0 + golden(number));
return sqrt (1.0 + golden(number - 1));
return (1.0 + golden(number - 1));
return (1.0 + golden(number));
	return sqrt (1.0 + golden(number - 1));

[2342] In 1735 Leonard Euler proved a remarkable result, which was the solution to the Basel Problem, first posed in 1644 by Pietro Mengoli. This result gave a simple expression for mc042-1.jpg. The formula states that mc042-2.jpgis equal to the limit, as n goes to infinity, of the series mc042-3.jpg. Can this series be computed recursively? Yes, but the code will be very long No, because the base case is not zero Yes No, because there is no base case	Yes

[2343] In 1735 Leonard Euler proved a remarkable result, which was the solution to the Basel Problem, first posed in 1644 by Pietro Mengoli. This result gave a simple expression
The formula states that equal to the limit, as n goes to infinity, of the series
 	
Which function below is a correct recursive implementation that approximates this infinite series?
	double computePI(int number)
	{
	if (number <= 1) { return 1.0;}
	return 1.0 / (number * number) + computePI(number - 1);
	}

expression for mc044-1.jpg. The formula states that mc044-2.jpgis equal to the limit, as n goes to infinity, of the series mc044-3.jpg. Which statement below is the correct base case for a recursive implementation that approximates this infinite series?
 	
if (number == 0) { return 1.0 / (number * number);}
if (number <= 1) { return 1.0;}
if (number <= 1) { return 0.0;}
if (number == 1) { return (number * number);}	

[2345] In 1735 Leonard Euler proved a remarkable result, which was the solution to the Basel Problem, first posed in 1644 by Pietro Mengoli. This result gave a simple expression for mc045-1.jpg. The formula states that mc045-2.jpgis equal to the limit, as n goes to infinity, of the series mc045-3.jpg. Which statement below is the recursive case for a recursive implementation that approximates this infinite series?
 	
return 1.0 / (number * number) + computePI(number - 1);
return 1.0 + computePI(number);
return 1.0 + computePI(number - 1);
return 1.0 / (number * number) + computePI(number);
	return 1.0 / (number * number) + computePI(number - 1);

[2346] One remarkably simple formula for calculating the value of is the so-called Madhava-Leibniz series: 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 greater than one
When the value that is returned from the function is zero
When the parameter variable is zero
	When the parameter variable is less than or equal to one

[2347] One remarkably simple formula for calculating the value of mc047-1.jpg is the so-called Madhava-Leibniz series: mc047-2.jpg = mc047-3.jpg . 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 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
 	
it is called just one time to set the sign of the final result
 	
it makes sure the sign (positive or negative) alternates as each term of the series is computed
	it makes sure the sign (positive or negative) alternates as each term of the series is computed

[2348] Assuming that you need to write a recursive function calc_prod(int n) to calculate the product of the first n integers, which of the following would be a correct way to simplify the input for the recursive call? Call calc_prod(n - 1) and multiply by n. Call calc_prod(n + 1) and multiply by n. Call calc_prod(n - 2) and multiply by n. Call calc_prod(1) and multiply by n.	Call calc_prod(n - 1) and multiply by n.
[2349] Suppose you need to write a recursive function power(double x, int n) that calculates x to the power of n. Which of the following would be a correct way to implement the function power? Call power(x, n) and multiply by (n - 1). Call power(x, n - 1) and multiply by n. Call power(x - 1, n) and multiply by x. Call power(x, n - 1) and multiply by x.	Call power(x, n - 1) and multiply by x.