

Total combine1

★ Leave the first rating

Terms in this set (1660)

Unix and C	Ken Thomson and Dennis Ritchie
Fortran	John Backus
Simula	O. Dahl & K. Nygaard
Berkeley Systems Distribution Unix	Bill Joy
C++	Bjarne Stroustrup
GNU, GCC and Free Software	Richard Stallman
Code is written in machine (and assembly) language for a specific processor; thus it is non-portable or machine dependent.	native code machine language

Which of these statements apply to C++?	<div>More efficient than Java or Python</div> <div>Produces native code that runs on the CPU</div> <div>Compiles to native code</div>
Converts processed source code to object code.	Compiler
Allows you to run your program in a controlled environment.	Debugger
Used by compiler to produce object code.	Assembler
Combines object modules to produce an executable.	Linker
Provides instructions for building your program.	Make
Reads an executable image on disk and starts it running.	Loader
Performs text substitution on your source code.	Preprocessor
<div>What is wrong with this IPO code fragment? cout << "Name: "; string name; cout << "Hello, " << name << endl; cin >> name;</div>	Input occurs after output
<div>>></div> <div>cout</div> <div><<</div> <div>cin</div> <div>\n</div> <div>endl</div>	<div>Extraction or input operator</div> <div>Analogous to Java's System.out</div> <div>Insertion or output operator</div> <div>Similar to Java's Scanner objects</div> <div>Escape character</div> <div>Stream manipulator</div>

<div>What kind of error is this?</div> <div>error: expected ';' after expression</div>	A syntax error
<div>What is the problem here?</div> <div>You have submitted another student's completion code</div>	You filled out the STUDENT variable incorrectly
<div>What is the problem here?</div> <div>make: *** No targets specified and no makefile found. Stop.</div>	The programmer is in the wrong directory.



<pre>int main() { }</pre>	
<p>Below is the main function from the f2c program in Chapter 1. Which line(s) contain a function declaration?</p> <pre>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</pre>	None of these
<p>Below is the main function from the f2c program in Chapter 1. Which line(s) uses the character input stream?</p> <pre>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</pre>	Line 17
<p>Below is the main function from the f2c program in Chapter 1. Which line(s) contain a function call?</p> <pre>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</pre>	Line 18
<p>Below is the main function from the f2c program in Chapter 1. Which line(s) use the insertion operator?</p> <pre>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</pre>	Line 19 Line 15
<p>Below is the main function from the f2c program in Chapter 1. Which line(s) contain a output statement?</p> <pre>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</pre>	Line 15 Line 19
<p>Below is the main function from the f2c program in Chapter 1. Which line(s) contain a variable defintion?</p> <pre>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</pre>	Line 16 line 18



<pre>#include <iostream> using namespace std; int main() { cout << "Hello, World"; } make example ./example</pre>	
What command only builds hw04?	make
What command checks hw04 for correctness?	make test
What command hands in hw04 for course credit?	make submit
What command makes hw the current folder?	cd ~/workspace/cs150/hw
all comebine	...
What does it mean for a function to have a polymorphic parameter?	The function accepts an object as a parameter - the object may be a base-class object or a derived-class object
Which of the following statements is true with regard to type tags? I. If a new class is added to the hierarchy, the type-tag code will likely need to be revised II. Virtual functions provide a cleaner, more extendible mechanism III. Type tags are never used by programmers	I, II
Suppose myfunction is defined as follows. How many calls to myfunction result, including the first call, when myfunction(9) is executed? int myfunction(int n) { if (n <= 2) { return 1; } return n * myfunction(n - 1); }	8
Which of the statements following the code snippet is true? <pre>// class CashRegister { public: CashRegister(); CashRegister(int count); void set_item_count(int count); void view() const; private: int item_count; }; CashRegister::CashRegister() { set_item_count(0); } CashRegister::CashRegister(int count) { set_item_count(count); }</pre>	CashRegister() is the default constructor because it has no parameters.



<pre>class Employee { public: Employee(string name) { this->name = name; } void Employee_info() { cout << "My name is " << name << ". "; } private: string name; }; class Manager : public Employee { public: Manager(string name) : Employee(name) { } void Employee_info() { cout << "I am a Manager."; Employee::Employee_info(); } }; int main() { Manager manager = Manager("Susan Allen"); manager.Employee_info(); return 0; }</pre>	
<p>What is true about the statement given below?</p> <pre>int* ptr_num;</pre>	<p>ptr_num contains the memory location of an integer variable.</p>
<p>Class Manager inherits from class Employee. Which of the following statements are true?</p>	<p>A Manager constructor can pass data to an Employee constructor</p>
<p>What is the output of the following code snippet?</p> <pre>// class CashRegister { public: void set_item_count(int count); void view() const; private: int item_count; }; void CashRegister::view() const { cout << item_count << endl; } void CashRegister::set_item_count(int count) { item_count = count; } int main() { CashRegister reg1, reg2; reg1.set_item_count(15); reg2.set_item_count(10); reg1.view(); reg2.view(); return 0; } //</pre>	<p>15 10</p>
<p>Suppose that you declare an array int num[10]. Assuming the function declaration statement given below, what would you use to pass the array to the given function?</p> <pre>int sum_array(int arr[]) { } </pre>	<p>sum_array(num)</p>
<p>Which of the following statements is correct about an accessor member function?</p>	<p>It returns the value of a data member of an object but does not modify any data member values.</p>
<p>When tracing the execution of a recursive function, what can remove confusion about the current point of execution when debugging the program?</p>	<p>examination of the call stack</p>
<p>What is the output of the following code snippet?</p> <pre>int arr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; int* ptr = arr; ptr = ptr + 5; cout << *ptr << endl;</pre>	<p>6</p>

Total combine1		Study	<div></div>
<pre>class Car { public: Car(); Car(double new_speed); double get_speed() const; private: double speed; }; class AeroCar : public Car { public: AeroCar(); AeroCar(double new_height, double new_speed); void display_data() const; private: double height; }; int main() { Car c1; Car c2(10); AeroCar ac1; Aerocar ac2(10, 20); c1 = ac1; c1 = c2; ac1 = ac2; ac1 = c1; return 0; }</pre>			
<p>In the following code snippet, which constructor is called for the object declaration CashRegister reg(5)?</p> <pre>// class CashRegister { public: CashRegister(); CashRegister(int count); void set_item_count(int count); void view() const; private: int item_count; }; CashRegister::CashRegister() { set_item_count(0); } CashRegister::CashRegister(int count) { set_item_count(count); } int main() { CashRegister reg(5); return 0; } //</pre>		<p>CashRegister(int count)</p>	

<p>Study the functions below:</p> <pre>int myfun1(int n) { if (n < 1) { return 0; } return myfun2(n * 2 - 1); } int myfun2(int n) { if (n == 1) { return 1;} return n + myfun2(n - 2); }</pre> <p>What does the function myfun1 compute for positive n?</p>		<p>Sum of the first n odd integers</p>	
<p>One way to determine the run-time efficiency of a recursive function is to</p>		<p>estimate the shape and size of its call tree</p>	
<p>Using pictures can help determine the structure of data and connections of data within a program. How are pointers usually represented in a diagram?</p>		<p>Data storage locations are represented as boxes; pointers are drawn as arrows with the tip pointing to the boxes</p>	

Total combine1		Study	1/1
<pre>int foo(int a[], int n) { if (n == 0) { return a[n]; } return { return foo(a, n - 1) + a[n]; } }</pre>			
<p>Which of the following is a legal statement to dynamically allocate memory for an array whose size is not known until run time?</p> <pre>int size; // Assume that size is set at run time</pre>		<pre>int* intarray = new int[size];</pre>	
<p>What does the new operator do in the following statement?</p> <pre>double* some_num = new double[20];</pre>		It allocates an array of size 20, and yields a pointer to the starting element.	
<p>Consider a situation where the function reverse_string should reverse a string that is passed as an argument. Which of the following options correctly completes the reverse_substring function?</p> <pre>string reverse_substring(string str, int start, int end) { if (start >= end) { return str; } char ch = str[start]; str[start] = str[end]; str[end] = ch; return // complete this statement } string reverse_string(string str) { return reverse_substring(str, 0, str.length() - 1); }</pre>		<pre>reverse_substring(str, start + 1, end - 1);</pre>	
<p>Consider the member function call</p> <pre>shpcrt.add_product(5, 59.75);</pre> <p>Which of the following describes the role of 5 and 59.75 in this call?</p>		They are explicit parameters.	

Total combine1		Study	<div></div>
<pre>class Building { public: Building() { name = ""; height = 0; } Building(string n, double h) { name = n; height = h; } void set_name(string n) { name = n; } void set_height(double h) { height = h; } string get_name() const { return name; } double get_height() const { return height; } private: string name; double height; }; class SkyScraper : public Building { public: SkyScraper() { width = 0; } SkyScraper(double w) { width = w; } SkyScraper(string n, double h, double w) : Building(n, h) { width = w; } void set_width(double w) { width = w; } void print_data() const; private: double width; }; void SkyScraper::print_data() const { cout << "Name: " << get_name() << "; Height: " << get_height() << "; Width: " << width << endl; } int main() { SkyScraper bldg1; SkyScraper bldg2(100); SkyScraper bldg3("World Trade Tower", 3000, 100); bldg2.print_data(); return 0; }</pre>			
<p>What is the output of the following code snippet?</p> <pre>double* temperature = NULL; cout << temperature << endl;</pre>		0	

<p>What does this function do?</p> <pre>bool myfunction(string s) { if (s.length() <= 1) { return true; } char first = s[0]; char last = s[s.length() - 1]; if (first == last) { string shorter = s.substr(1, s.length() - 1); return myfunction(shorter); } else { return false; } }</pre>		The code snippet returns true only for strings of any length consisting of the same character.	
<p>What does class aggregation mean?</p>		It means that an object of one class acts as the data member of an object of another class.	
<p>Which of the following statements is correct about a recursive function?</p>		A recursive function calls itself.	
<p>The Manager class inherits from the Employee base class. The Manager class overrides the get_salary()function. What is wrong with the following definition of get_salary() in the Manager class?</p> <pre>double Manager::get_salary() const { double base_salary = get_salary(); return base_salary + bonus; }</pre>		The call to get_salary should be written as Employee::get_salary();	
<p>Two quantities a and b are said to be in the golden ratio if (a+b)/a is equal to a/b. 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 = sqrt(1+sqrt(1+sqrt(1+sqrt(1+sqrt(1+... If the function double golden (int) is a recursive implementation of this function, what should be the recursive call in that function?</p>		return sqrt (1.0 * golden(number - 1));	

Total combine1		Study	<div><div></div><div></div><div></div></div>
Which of the following code snippets is legal for implementing the CashRegister class member function named add_item?	<div>.</div> <div>item_count = item_count + 1; total_price = total_price + price; }</div>		
<div>Consider the following code snippet:</div> <div><pre>class Question { public: Question() { text = ""; answer = ""; } virtual void set_text(string s) { cout << "Question : set_text function" << endl; } private: string text; string answer; }; class ChoiceQuestion : public Question { public: ChoiceQuestion() {} void set_text(string s) { cout << "ChoiceQuestion : set_text function" << endl; } }; int main() { Question* q1 = new Question; ChoiceQuestion* cq1 = new ChoiceQuestion; q1 = cq1; q1->set_text("Which function?"); return 0; }</pre></div> <div>Which of the following is true about the statement "q1->set_text("Which function?");"?</div>	<div>ChoiceQuestion::set_text(string new_text) is called by q1</div>		
<div>What is output of the code snippet given below?</div> <div><pre>string name = "Oscar DeGama"; cout << name[7] << name[8] << name[9] << endl;</pre></div>	<div>eGa</div>		
<div>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 pi. The formula states that pi^2/6 is equal to the limit, as n goes to infinity, of the series 1/1 + 1/2^2 +... + 1 / n^2. Which statement below is the recursive case for a recursive implementation that approximates this infinite series?</div>	<div>return 1.0 / (number * number) + computePI(number - 1);</div>		
<div>Suppose the class Manager is derived from the class Employee. Consider these statements:</div> <div><pre>Employee* pe = new Employee; Manager* pm = new Manager; pe = pm;</pre></div> <div>What happens at the "pe = pm" assignment?</div>	<div>A derived-class pointer is assigned to a base-class pointer</div>		
<div>Two quantities a and b are said to be in the golden ratio if (a+b)/a is equal to a/b. 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 fraction: golden ratio =1 + 1/(1 + 1/(1 + 1/(1 + 1/.... Which function below is a correct recursive implementation of this continued fraction?</div>	<div><pre>double golden(int number) { if (number <= 1) { return 1.0;} return 1.0 + 1.0 / golden(number - 1); }</pre></div>		
<div>Based on the following code snippet, which of the following function calls are legal? Assume that car is a Car object and aero is an AeroCar object.</div> <div><pre>class Car { public: Car(); void set_speed(double new_speed); double get_speed() const; void display() const; private: double speed; }; class AeroCar : public Car { public: AeroCar(); void set_speed(double new_speed); void set_height(double new_height); double get_height() const; private: double height; };</pre></div>	<div>car.get_speed() and aero.get_speed()</div>		

Total combine1		Study	<div>100%</div>
<p>to CashRegister objects. Given the declaration of the array all_registers below, which code snippet correctly calls the clear() function on every object pointed to from the all_registers array? ?</p> <pre>// CashRegister* all_registers[20]; //</pre>	<pre>all_registers[i]->clear(); }</pre>		
<p>You are given the class definition for CashRegister, and you are writing client code to set up an array of pointers to CashRegister objects. Given the declaration of the array below, which code snippet correctly allocates CashRegister objects from the heap for the entire array?</p> <pre>// CashRegister* all_registers[20]; //</pre>	<pre>for (int i = 0; i < 20; i++) { all_registers[i] = new CashRegister; }</pre>		
<p>Suppose that we have a Question class that contains two data members - a query and an answer string. The NumericQuestion class inherits from Question. Which of the following is true?</p>	<p>NumericQuestions contains both a query and an answer string.</p>		
<p>The technique for hand-tracing objects puts public member functions on the front of an index card, and the private data members on the back. Why?</p>	<p>Data members are hidden from outside client code and this simulates the principle of encapsulation</p>		
<p>A word-unit palindrome is a sentence that produces the same words forward or backward. For example, "Fall leaves after leaves fall" is a word-unit palindrome. Suppose you wish to implement a recursive function that can check sentences to see if they are word-unit palindromes. What is the first step?</p>	<p>break the input into smaller parts that can themselves be inputs to the problem</p>		
<p>In the following class definition, which of the data members or member functions are hidden from the class users?</p> <pre>. class Car { public: Car(); void start(); void stop(); private: double speed; void set_speed(double new_speed); };</pre>	<p>double speed and void set_speed(double new_speed)</p>		
<p>What does the function f do?</p> <pre>struct Point2D { double x; double y; }; struct Triangle { Point2D v1; Point2D v2; Point2D v3; }; void f(Triangle& t) { int temp = 12.5; temp = tv1.x; tv1.x = tv1.y; tv1.y = temp; } int main() { Triangle mytri; mytri.v1.x = 1.0; mytri.v1.y = 22.5; f(mytri); }</pre>	<p>Swaps values of x and y in vertex 1 of an argument of type Triangle</p>		



<pre>// class Building { public: Building(); Building(int new_height); void set_height(int new_height); void view() const; private: int height; }; Building::Building() { set_height(0); } Building::Building(int new_height) { set_height(new_height); } int main() { Building bldg(500); return 0; } //</pre>	
<p>In the given code snippet, what type of member function is view()?</p> <pre>// class CashRegister { public: void view() const; private: int item_count; double total_price; }; void CashRegister::view() const { cout << item_count << endl; cout << total_price << endl; } //</pre>	<p>Accessor member function</p>
<p>What is the value of the return variable m in the following code snippet, if a = {1, 2, 3, 4, 5, 6} and n = 5?</p> <pre>int myfunction(int a[], int n) { if (n == 1) { return a[0]; } int m = myfunction(a, n - 1); if (a[n] > m) { return a[n - 1]; } else { return m; } }</pre>	<p>5</p>
<p>Which of the following is a benefit of encapsulation?</p>	<p>It guarantees that an object cannot be accidentally put into an inconsistent state.</p>
<p>Consider the code snippet below</p> <pre>int important_calculation(int size) { int the_answer = 42; int* ptr = new int[size]; ptr[0] = the_answer; return the_answer; }</pre>	<p>A. There is a new without a matching delete. B. There is an uninitialized pointer that is used. CorrectC. Both b and c. D. There is memory that is never reclaimed.</p>



<pre>class Home { public: Home() { address = ""; landmark = ""; } virtual void set_address(string address) { cout << "Home : set_address function" << endl; } private: string address; string landmark; }; class Villa : public Home { public: Villa() {} void set_address(string address) { cout << "Villa : set_address function" << endl; } }; int main() { Home* q1 = new Home; Villa* cq1 = new Villa; q1 = cq1; q1->set_address("Which function?"); return 0; }</pre> <p>Which of the following is true about the statement "q1->set_address("Which function?");"?</p>	
<p>What is the output of the following code snippet?</p> <pre>int myfunction(int n) { if (n < 2) { return 1; } return n * myfunction(n - 1); } int main() { cout << myfunction(3) << endl; return 0; }</pre>	6
<p>Consider the constructor for a derived class. Because a derived-class constructor can only initialize the data members of the derived class, how can it initialize base-class data members via a constructor other than the default constructor?</p>	The derived-class constructor can supply arguments to the base-class constructor via the base-class initializer
<p>In the following code snippet, what is the return value when the parameter values are m = 10 and n = 4?</p> <pre>int foo(int m, int n) { if ((m == n) { return n; } else if (m < n) { return foo(m, n - m); } else { return foo(m - n, n); } }</pre>	2
<p>Given the following declaration of the variables p1 and p2, which code fragment prints "Hello" if the value of x in the variable p1 is larger than the value of x in the variable p2?</p> <pre>struct Point2D { double x; double y; }; Point2D p1; Point2D p2;</pre>	<pre>-- if (p1.x > p2.x) { cout << "Hello"; }</pre>
<p>Consider the following code snippet:</p> <pre>int* num = new int; *num = 10; cout << num << endl; delete num; num = num * 2; cout << num << endl;</pre> <p>Which of the following statements is correct?</p>	The num pointer is being used after the memory it is pointing to has been deleted.



```
#include <iostream>
using namespace std;

class Car
{
public:
    Car(double speed);

    void start();
    void accelerate(double speed);
    void stop();
    double get_speed() const;

private:
    double speed;
};

Car::Car(double speed)
{
    this->speed = speed;
}

void Car::start()
{
    speed = 1;
}

void Car::accelerate(double speed)
{
    this->speed = this->speed + speed;
}

void Car::stop()
{
    speed = 0;
}

double Car::get_speed() const
{
    return speed;
}

int main()
{
    Car* c1 = new Car(90);
    c1->start();
    c1->accelerate(90);
    cout << "Speed: " << c1->get_speed() << endl;
    c1->stop();
    return 0;
}
//
```

Which of the following options denotes the newline character?

|

'\n'

What is displayed when you execute the following code snippet?

```
int ch = 100;
cout << &ch << endl;
```

The memory location of ch



<pre>#include <iostream> using namespace std; int find(int a, int b) { int val; if (a > b) { val = a; } else { val = b; } return val; } int find(int a[], int size) { int val; if (size == 1) { val = a[0]; } else { val = find(a[size - 1], find(a, size - 1)); } return val; } int main() { int a[7] = { 6, 5, 4, 3, 9, 2, 1 }; cout << find(a, 7); return 0; }</pre>	
<p>Study the following class interface for the class AeroPlane:</p> <pre>// class AeroPlane { public: void set_new_height(double new_height); void view() const; void view_new_height() const; AeroPlane(); AeroPlane(double new_height); AeroPlane(double new_height, double new_speed); AeroPlane(int new_height, int new_speed); private: double height; double speed; }; //</pre> <p>Which of the following constructors is called for the object declaration AeroPlane c1(10, 100)?</p>	<pre>AeroPlane(int new_height, int new_speed)</pre>
<p>Consider the code snippet below.</p> <pre>int ch = 100;</pre> <p>Which of the following is a legally correct way of obtaining the memory location of ch and printing it (the memory location) to standard output, based on the given code snippet?</p>	<pre>cout << &ch << endl;</pre>
<p>When designing classes to solve a specific problem, one simple approach to discovering classes and member functions is to</p>	<p>identify nouns and verbs, which correspond to classes and functions, respectively</p>



<pre>// #include <iostream> using namespace std; class Cheetah { public: void set_speed(double new_speed); private: double speed; }; void Cheetah::set_speed(double new_speed) { speed = new_speed; } int main() { Cheetah ch1; ch1.set_speed(144); cout << "Cheetah speed: " << ch1.speed; return 0; } //</pre>	
<p>Which location of the array arr does ptr point to right after you assign an array to a pointer variable, as shown in the following code snippet?</p> <pre>int arr[10]; int* ptr = arr;</pre>	arr[0]
<p>The AeroCar class inherits from the Car class and overrides the set_speed(double new_speed) function. In the AeroCar set_speed function, how can the Car set_speed function be called?</p>	Car::set_speed(new_speed)
<p>What is looked for in the problem description when determining the name of a class?</p>	Nouns
<p>Study the functions below:</p> <pre>int myfun1(int n) { if (n < 1) { return 0; } return myfun2(n * 2 - 1); } int myfun2(int n) { if (n == 1) { return 1;} return n + myfun2(n - 2); }</pre> <p>What is the output for the statement cout << myfun1(-7)?</p>	0
<p>Consider the following class interfaces:</p> <pre>class Teacher { public: Teacher(); Teacher(string new_name); string get_name() const; private: string name; }; class MathsTeacher : public Teacher { public: MathsTeacher(); MathsTeacher(string new_qualification, string new_name); void display_data() const; private: string qualification; }; int main() { Teacher t1, t2("John"); MathsTeacher mt1, mt2("TopLevel", "Sarah"); t1 = mt1; t1 = t2; mt1 = mc2; mt1 = t1; return 0; }</pre> <p>Which one of the preceding assignment statements in the function main() would slice away the data of the derived class object?</p>	t1 = mt1



How can a derived class override a base class function?	By providing a new implementation for a function with the same name and parameter types
The reserved word virtual is used with a base-class member function to alert the C++ compiler that it must determine the actual member function to call at run-time. Thus derived classes can have their own (possibly completely different) versions of the base-class function. When a base-class defines a member function to be virtual, what should each derived class do?	Each derived class should over-ride the virtual base-class function so that it defines a function with the same name that is unique to the derived class
<div>What is the output of the following code snippet?</div> <div><pre>#include <string> #include <iostream> using namespace std; string reverse(string str, int start, int end) { if (start >= end) { return str; } char ch = str[start]; str[start] = str[end]; str[end] = ch; return reverse(str, start + 1, end - 1); } string string_reverse(string str) { int index = str.length() / 2; string str1 = reverse(str, 0, index - 1); string str2 = reverse(str, index, str.length() - 1); string str3 = str1.substr(0, index); string str4 = str2.substr(index, str.length() - index); return str3.append(str4); } int main() { cout << string_reverse("1234") << endl; return 0; }</pre></div>	2143
<div>What is wrong with the following code snippet?</div> <div><pre>void myfun(char* p) { p = toupper(p); } int main() { string myword = "YouHadMeAtHello"; for (int i = 0; i < 10; i++) { myfun(myword[i]); } cout << myword << endl; return 0; }</pre></div>	Must use c_str() to access individual characters of the string variable myword
<div>Study the following code snippet:</div> <div><pre>class Employee { public: Employee(); Employee(string new_name); Employee(double new_salary); Employee(string new_name, double new_salary); private: string name; double salary; }; class Manager : public Employee { public: Manager(); Manager(string new_department, string name, double salary); private: string department; };</pre><div>Which among the following is the legal way of implementing the constructor of the Manager class that passes parameters to a base-class constructor?</div></div>	<pre>Manager::Manager(string new_department, string new_name, double new_salary) : Employee(new_name, new_salary) { department = new_department; }</pre>

Total combine1		Study	<div>1/1</div>
<div><pre>class Car { public: Car(); void set_speed(double new_speed); double get_speed() const; private: double speed; };</pre></div> <div>The AeroCar class inherits from the Car class. For the AeroCar class to override the set_speed function, what must be done?</div>			
<div>What is the output of the code snippet given below?</div> <div><pre>int arr[5] = {1, 2, 3, 4, 5}; int* ptr = arr; cout << "arr[0] contains a value of " << *ptr << endl;</pre></div>	<div>arr[0] contains a value of 1</div>		
<div>Describe how the two arrays index and data are related to one another after the code snippet below executes:</div> <div><pre>int* index[5]; int data[10] = {4, 8, 1, 3, 5, 9, 3, 2, 6, 0}; int i = 0; int* p = &data[9]; for (int i = 0; i < 5; i++) { index[i] = p; p--; }</pre></div>	<div>The elements of index point to the elements of data as follows: index[0] points to data[9] index[1] points to data[8] etc. for all five elements of index</div>		
<div>Which location of the array num does ptr point to right after you assign an array to a pointer variable, as shown in the following code snippet?</div> <div><pre>int num[5]; int* ptr = num;</pre></div>	<div>num[0]</div>		
<div>Consider the following recursive function:</div> <div><pre>1. int triangle_area(int side_length) 2. { 3. if (side_length <= 0) { return 0; } 4. if (side_length == 1) { return 1; } 5. int smaller_side_length = side_length - 2; 6. int smaller_area = triangle_area(smaller_side_length); 7. return smaller_area + side_length; 8. }</pre></div> <div>What is returned when this function is called with a value for side_length of 7?</div>	<div>16</div>		
<div>Which of the following statements is true about data members of a class definition?</div>	<div>Every object of the defined class has its own set of data members, with possibly different values.</div>		
<div>Which symbol in C++ indicates inheritance?</div>	<div>:</div>		
<div>One remarkably simple formula for calculating the value of pi is the so-called Madhava-Leibniz series: $\pi/4 = 1 - 1/3 + 1/5 - 1/7 + 1/9 - \dots$. Consider the recursive function below to calculate this formula:</div> <div><pre>double computePI(int number) { if (number <= 1) { return 1.0;} int oddnum = 2 * number - 1; return computesign(number) * 1.0 / oddnum + computePI(number - 1); }</pre></div> <div>In this recursive function, what is the role of the helper function computesign?</div>	<div>it makes sure the sign (positive or negative) alternates as each term of the series is computed</div>		
<div>What does this function do?</div> <div><pre>bool foo(string s) { if (s.length() <= 1) { return true; } char first = s[0]; char last = s[s.length() - 1]; if (first != last) { string shorter = s.substr(1, s.length() - 1); return foo(shorter); } else { return false; } }</pre></div>	<div>The code snippet returns false if the string is a palindrome.</div>		

Total combine1		Study	
I. An empty string II. A shape without any area III. A negative value for time			
Which statement, if executed immediately after the given one, creates orphaned heap memory? CashRegister* register_pointer = new CashRegister;	register_pointer = new CashRegister;		
A word of any length is a palindrome if its characters are the same forward and in reverse. Which of the following simplifications is helpful in designing a recursive solution to the problem of checking to see if a word is a palindrome?	The word obtained by removing the first and last letters is a palindrome, and the first and last letters are the same.		
Which of the following hides the implementation details of the data members and member functions within a class?	Encapsulation		
Consider the two functions that are defined here. Which of the options are true about these functions? I. The function f1 is a recursive solution, and the function f2 is an iterative solution II. The two functions execute at about the same speed III. For a string of length n, the function f1 makes about n/2 recursive calls but the function f2 performs about n/2 iterations string f1(string str) { string val = ""; if (str == "" str.length() <= 1) { return str; } string first = str.substr(0, 1); string middle = str.substr(1, str.length() - 2); string last = str.substr(str.length() - 1, 1); return return last + f1(middle) + first; } string f2(string str) { int len = str.length(); int mid = len / 2; for (int i = 0; i < mid; i++) { char ch = str[i]; int pos = len - i - 1; str[i] = str[pos]; str[pos] = ch; } return str; }	I, II, and III		
How should base-class data members be referenced and used from derived classes?	The derived class should use base-class accessors and mutators as a way to access or update base-class data members		
Discovering aggregation relationships between classes can help the process of implementation because	one object may need to contain objects of another class as data elements in order to simplify the problem solution		
The Department of Motor Vehicles uses a vehicle registration program that declares a Vehicle class as a base class. The Car class and the Truck class both inherit from the Vehicle class. Which types of objects can be passed to the function register(Vehicle& v)?	Vehicle, Car and Truck objects		
Suppose you are to write a recursive function to calculate the "area" of a triangle shown as follows, where the area of each □ square is 1: □□□□□□□□ □□□□□□□ □□□□□□ □□□□□ □□□□ □□□ □□ □ Assume the base of the triangle (top edge) always has an odd number of □ squares, and each subsequent line has two fewer squares (and hence the triangle always ends with a point that has just one □ square). Consider the recursive function below: 1. int triangle_area(int side_length) 2. { 3. if (side_length <= 0) { return 0; } 4. if (side_length == 1) { return 1; } 5. int smaller_side_length = side_length - 1; 6. int smaller_area = triangle_area(smaller_side_length); 7. return smaller_area + side_length; 8. } Will this function correctly compute the area of triangles as shown above for odd side length?	no, because line 5 should be int smaller_side_length = side_length - 2;		

Total combine1		Study	<div><div></div><div></div><div></div></div>
<div><div><div>int* num = new int; *num = 10;</div></div></div>			
<div><div><div>Consider the code snippet below: int main() { Shape* shapes[NUM_OBJECTS]; shapes[0] = new Circle(0, 0, 100, 150); shapes[1] = new Rectangle(200, 200, 50, 100); shapes[2] = new Circle(300, 50, 250, 250); shapes[3] = new Rectangle(100, 350, 200, 150); for (int i = 0; i < NUM_OBJECTS; i++) { shapes[i]->draw(draw_area); } for (int i = 0; i < NUM_OBJECTS; i++) { delete shapes[i]; } }</div><div>In order for pointers to the two different objects Circle and Rectangle to be put into the same array (shapes), what must be true?</div></div></div>	<div>The objects must be related to one another via inheritance</div>		
<div><div><div>Which of the following statements is correct about the public interface for the Car class? . class Car { public: void start(); private: double speed; void stop(); };</div></div></div>	<div>CorrectA. All of the listed items. B. The code snippet includes the public interface for the class but lacks the definition of the member functions. C. This interface does not contain the stop() function. D. This interface contains the start() function.</div>		
<div><div><div>What is the output of the following code snippet? void myfun(char* p) { p = toupper(p); } int main() { char myword[20] = "YouHadMeAtHello"; for (int i = 0; i < 10; i++) { myfun(&myword[i]); } cout << myword << endl; return 0; }</div></div></div>	<div>YOUHADMEATHello</div>		
<div><div><div>Consider the code snippet below, which uses two pre-defined objects Time and ExtTime: int main() { Time* test_time[SIZE]; int rand_hrs, rand_mins, rand_secs; ZoneType rand_zone; for (int i = 0; i < SIZE; i++) { rand_hrs = rand() % 3600; rand_mins = rand() % 60; rand_secs = rand() % 60; rand_zone = (ZoneType)(rand() % 8); if (i % 2) { test_time[i] = new ExtTime(rand_hrs, rand_mins, rand_secs, rand_zone); } else { test_time[i] = new Time(rand_hrs, rand_mins, rand_secs); } } for (int i = 0; i < SIZE; i++) { test_time[i]->write(); cout << endl; } }</div><div>In order for pointers to objects of two different classes, Time and ExtTime, to be put into the same array (test_time), what must be true?</div></div></div>	<div>The objects must be related to one another via inheritance</div>		
<div><div><div>What is the relationship between a base class and a derived class called?</div></div></div>	<div>Inheritance</div>		

Total combine1		Study	<div>1/1</div>
<pre>#include <vector> #include <string> #include <iostream> using namespace std; vector<string> substrings(string str) { vector<string> vec; if (str == "" str.length() <= 1) { vec.push_back(str); return vec; } string first_char = str.substr(0, 1); string rest = str.substr(1, str.length() - 1); vector<string> subs = substrings(rest); int size = subs.size(); for (int i = 0; i < size; i++) { string s = first_char.substr(0, 1); s.append(subs[i]); subs.push_back(s); } return subs; } int main() { vector<string> vec = substrings("432"); for (int i = 0; i < vec.size(); i++) { cout << vec[i] << endl; } return 0; } //.</pre>	<div>42</div> <div>432</div>		
Some programmers are careful about using this within member functions to make clear the distinction between	data members and other variables		
<p>What is the reason for including the following code snippet in the header file Car.h?</p> <pre>#ifndef CAR_H #define CAR_H class Car { public: Car(); Car(double speed); void start(); void accelerate(double speed); void stop(); double get_speed() const; private: double speed; }; #endif</pre>	To define the interface for the Car class		
<p>Based on the following code snippet, which of the following function calls are legal? Assume that hme is a Home object and vil is a Villa object.</p> <pre>class Home { public: Home(); void set_location(string new_location); string get_location() const; void display() const; private: string location; }; class Villa : public Home { public: Villa(); void set_location(string new_location); void set_size(double new_size); double get_size() const; private: double size; };</pre>	hme.get_location() and vil.get_location()		

Total combine1		Study	<div>100%</div>
out all possible permutations of the cards in a hand. What would be the base case?			
The destructor is a special member function and is invoked under which circumstance?		When a heap object is explicitly deleted with the delete command	
Consider a situation where you need to write a recursive function void reverse that reverses a string. Suppose your recursive solution removes the first character, reverses the string consisting of the remaining text, and combines the two. Which of the following would be a technique that may produce an easier solution?		Introduce a helper function that reverses a substring of the original string.	
<p>Consider the code snippet below, where the Circle and Rectangle objects are both derived classes from the base class Shape:</p> <pre>int main() { Shape* shapes[NUM_OBJECTS]; shapes[0] = new Circle(0, 0, 100, 150); shapes[1] = new Rectangle(200, 200, 50, 100); shapes[2] = new Circle(300, 50, 250, 250); shapes[3] = new Rectangle(100, 350, 200, 150); for(int i = 0; i < NUM_OBJECTS; i++) { // Code calling the draw function HERE } }</pre> <p>Which statement below could correctly be inserted in the code above where noted in order to invoke the draw function on the objects referenced by the shapes[] array?</p>		shapes[i]->draw();	
Suppose that we have a function that registers a Cycle object. We also have a Scooter object that is a specialized Cycle (defined by inheritance). The substitution principle states _____.		The Scooter object can be used in the Cycle registration function because it is a kind of Cycle.	
Consider a situation where you have written two different functions, both of which determine whether a string is a palindrome. The first function is a recursive function that calls itself n/2 times for a string of length n. The second function uses an iterative solution that loops n/2 times for a string of length n. Which of the following statements are true about this situation?		The iterative solution tends to be a little bit faster than the recursive solution.	
Which of the following is true about using recursion?		A recursive computation solves a problem by calling itself with simpler input.	
<p>Consider the member function call</p> <pre>shpcrt.add_product(5, 59.75);</pre> <p>Which of the following describes the role of shpcrt in this call?</p>		It is an implicit parameter.	
In order to support polymorphism, the virtual reserved word must be used with _____.		the base-class	
<p>Why is it important to write C++ code that is split into separate source files?</p> <p>I. Objects in C++ can only be defined using separate source files</p> <p>II. It is more efficient to compile since only files that have changed need to be recompiled</p> <p>III. It lessens the problem of multiple team members needing to simultaneously edit the same source file</p>		II, III	

Total combine1		Study	<div>...</div>
<pre>class Car { public: void start(); void accelerate(double acc_speed); void stop(); double get_speed() const; Car(); private: double speed; }; Car::Car() { speed = 0; } void Car::start() { accelerate(get_speed() + 10); } void Car::stop() { speed = 0; } void Car::accelerate(double acc_speed) { speed = speed + acc_speed; } double Car::get_speed() const { return speed; } int main() { Car c1; c1.start(); c1.accelerate(10); c1.get_speed(); c1.stop(); return 0; }</pre>			
<p>What is the output of the following code snippet?</p> <pre>char name[] = "Harry Houdini"; name[3] = 'v'; cout << name << endl;</pre>	Harvy Houdini		
<p>What is wrong with the code snippet below?</p> <pre>void did_it() { int* number1 = new int[10]; int* number2 = new int[20]; number1[0] = 100; number2[0] = number1[0]; number1 = number2; // more important stuff here delete[] number2; }</pre>	The memory that number1 points to originally is never released.		
<p>What is the output of the following code snippet?</p> <pre>int* ptr; ptr = ptr + 5; cout << *ptr << endl;</pre>	It results in an unpredictable error when the code is run because it uses an uninitialized pointer		
In C++, any class can be considered an exception class.	True		
The function <code>__</code> can check whether an expression meets the required conditions; if the conditions are not met, it terminates the program.	assert		
C++ provides all the exception classes you will ever need.	False		
A(n) <code>__</code> is an occurrence of an undesirable situation that can be detected during program execution.	exception		
Which of the following is a valid C++ statement?	assert(divisor != 0);		
The try block is followed by one or more <code>__</code> blocks.	catch		
The class <code>__</code> is designed to deal with illegal arguments used in a function call.	invalid_argument		



Which of the following statements creates a new exception class?	class myClass {};
When division by zero occurs and the problem is not addressed, the program crashes with an error message that is __ dependent.	IDE
Which of the following statements throws a valid exception in C++?	throw 2;
Which of the following options should you choose when an exception occurs in the program that analyzes an airline's ticketing transactions?	Log the error and continue.
The order of the catch blocks does not affect the program	False
In a sequence of try/catch blocks, the last catch block of that sequence should be ____.	catch(...){ }
A catch block specifies the type of exception it can catch and immediately terminates the program.	False
To use the assert function in your program, you should include the statement ____.	#include <cassert>
The class ____ is designed to deal with errors that can be detected only during program execution.	runtime_error
A catch block can have, at most, ____ catch block parameter(s).	one
The logic_error and runtime_error classes are defined in the header file ____.	stdexcept
If you want to include members in your exception class, you typically include the function ____.	what
The first step in problem solving is?	To understand the problem and its inputs and outputs
Which one of the following is a correct method of defining and initializing an integer variable with name value?	int value = 30;
Which of the following statements replaces input into the variable "value"?	cin >> value;
What is the meaning of x = 0; in C++?	It sets the variable x to zero
This line of code '#include <iostream>' is handled by:	The preprocessor
You can think of C++ as the merging of:	Simula and C
<pre>#include<iostream> int main() { cout << "Hello CS 150"; }</pre>	Does not compile
<pre>#include <iostream> using namespace std; int main() { cout << sqrt(64) << endl; }</pre>	Compiles on some platforms but incorrect
The assignment operator	Places a new value into a variable
What is the output of the following code snippet? <pre>#include <iostream> using namespace std; int main() { int value = 3; value++; cout << value << endl; return 0; }</pre>	4
Which one of the following operators computes the remainder of an integer division?	%
What statement is true about identifiers (variable and function names) in C++?	They may contain an underscore (_)
Assume you have a string variable str. Which is the correct way to find the number of characters it contains?	str.length();
Which of the following statements gives the absolute value of the floating point variable x? double x = -25.5;	abs(x);

Total combine1		Study	<div>1/1</div>
I. there is more than one integer type. II. The data type float(generally) uses twice the storage of type double III. Numeric ranges are typical but not guaranteed to be the same between compilers.			
What is result of evaluating the following expression? (45 / 6) % 5	2		
I. Although not required, constants are commonly named using uppercase letters II. Only integer values can appear as constants III. A variable defined with an initial value and using the modifier const cannot be accidentally changed IV. A named constant makes computations clearer Which of these are true?	I, III, IV		
Int sum = 22; sum = sum + 2; cout << sum++; // sum = sum + 4; What is printed when this runs?	2425		
int size = 42; cost = 9.99; cout << "size=" << size << ", cost=" << cost << endl; What prints out here(assuming all includes, etc).	size=42, cost=9.99		
extern int cost; int size = 42; cost = 9.99; cout << "size=" << size << ", cost=" << cost << endl; What prints out here(assuming all includes, etc).	...		
double bottles; double bottleVolume = bottles * 2; cout << bottleVolume << endl; What prints out here (assuming all includes, etc.)	Unpredictable result; logic error		
double bottles = 7; int num = 2.3; char ch = 7.5L;	Possible compiler warning if requested		
int x = 20, y = 10; x = (x - y) * 2; y = x--; What values are in x and y after running this code?	x -> 19, y -> 20		
int a; double b; string c; cin >> a >> b >> c; cout << a << ", " << b << ", " << c; input: 7 3.5 Steve Gilbert <-' What prints?	7, 3.5, Steve		
int a; double b; string c; cin >> a >> b >> c; cout << a << ", " << b << ", " << c; input: 7 <-' 3.5 <-' Steve Gilbert <-' What prints?	7, 3.5, Steve		
int a; double b; string c; cin >> a >> b >> c; cout << a << ", " << b << ", " << c; input: 3.5 7 Steve Gilbert <-' What prints?	3, 0.5, 7		
string firstname = "William", lastname; cout << lastname + ", " + firstname;	, William		
string name; name = "Shakespeare" + " , " + "William"; cout << name << endl; What prints? (assume all includes, etc.)	Compiler error on line 2		
int x = 2, y = 7; string result = "x=" + ", y=" + y; cout << result << endl; What prints? (assume all includes, etc.)	Compiler Error on line 2: can't concatenate ints		



Before using the C++ library type string, the program must include the header file ____.	<string>
What is the error in this code snippet?	Logic error: null statement in if body
<pre>string str1 = "abc"; string str2 = "xyz"; string str3 = str1 + '-' + str2;</pre> <p>What is stored in the variable str3?</p>	Does not compile
<p>What is the output of this code?</p> <pre>int num1 = 40; if (num1 <= 40) { cout << "F"; } if (num1 <= 75) { cout << "C"; } if (num1 <= 90) { cout << "B"; }</pre>	FCB
<pre>string square(int a) { return "Commencing"; }</pre> <p>Which option represents a legal call to the function named square()?</p>	string a = square(4);
<p>What is the problem with the following if statement?</p> <pre>double count = 15.0; if (count / 3.0) cout << "count is " << count << endl;</pre>	Compiles but condition not a Boolean expression
<p>Which condition(...) will protect against divide by 0?</p> <pre>if (...) { result = grade / num; cout << "Avoided divide by zero!" << endl; }</pre>	(num != 0)
<p>What is the problem with this if statement?</p> <pre>double avg = (g1 + g2 + g3 + g4) / 4.0; if (avg == 90.0) cout << "You earned an A!" << endl;</pre>	Using == to test doubles is error prone
<p>What is the output of this code snippet?</p> <pre>string str1 = "her", str2 = "cart"; if (str1 < str 2) cout << str2; else cout << str1;</pre>	her
<p>What prints? (assume all includes, namespace OK)</p> <pre>int a = 7; bool b = a - 2; cout << "b is " << b << endl;</pre>	b is 1
<p>What prints? (assume all includes, namespace OK)</p> <pre>int a = 3; if(a = 4) cout << "a is 4; weird!\n"; else cout << "a not 3\n";</pre>	a is 4
<p>What prints? (assume all include, namespace OK)</p> <pre>int a = 3; if (a == 3) a = 4; else; a = 5; cout << "a is " << a << "\n";</pre>	a is 5



<pre>int a = 3; if (a == 3) a = 4; a = 5; else a = 6;</pre>	
<p>"Give a 25% discount to children under 13 and to those who are 65 and older". What prints?</p> <pre>int age = 12; if (age < 13 && age > 65) cout << "25% discount" << endl; else cout << "No discount" << endl;</pre>	No discount
<p>"If the patron's is older than 12 or younger than 66, charge them 8.75 for the ticket. Charge 5.75 for everyone else." What prints?</p> <pre>int age = 12; if (age > 12 age < 66) cout << "\$ 8.75" << endl; else cout << "\$ 5.75";</pre>	\$8.75
<p>What prints? (Assume all includes, etc.)</p> <pre>string grade = "C"; if (grade == "A" "A+" "A-") cout << "Got an A\n"; else if (grade == "B" "B+" "B-") cout << "Got a B\n"; else if (grade == "C" "C+" "C-") cout << "Got a C\n";</pre>	"Got an A\n";
<p>Some if statements with placeholders for bodies. Which are executed?</p> <pre>if (2 < 3) S1; if (2) S2; if (0 == 0) S3; if (0) S4;</pre>	s1, s2, and s3
<p>What prints here?</p> <pre>cout << "Hi" cout << "Bye" << endl;</pre>	Hi
<p>What prints here?</p> <pre>cout << "Hi" && cout << "Bye" << endl;</pre>	HiBye
<p>What prints here?</p> <pre>!(cout << "Hi") && cout << "Bye" << endl;</pre>	Hi
<p>Assum x is an int with the value 4. What prints?</p> <pre>if (x <= 2) { if (x == 4) cout << "one"; } else cout << "two";</pre>	two
<p>Assume x is an int with the value 4. What prints?</p> <pre>if (x <= 2) if (x == 4) { cout << "one"; } else cout << "two";</pre>	Nothing. Runs but prints nothing.
<p>Which int values for x print "hi"?</p> <pre>switch (x - 3) { case 7: break; case 6: case 4: cout << "hi"; }</pre>	9 or 7
<p>How many times will the following loop run?</p> <pre>int i = 0; while (i < 9) { cout << i << endl; i++; }</pre>	9

Total combine1		Study	
<pre>bool token = false; while (token) cout << "Hello World!" << endl;</pre>			
<p>How many times does this display "Hello World"?</p> <pre>int i = 10; while (i >= 0) { cout << "Hello World" << endl; i--; }</pre>	11		
<p>Suppose str = "xyzw";. After the statement str.at(2) = 'Y'; The value of str is "___".</p>	xyYw		
<p>Given the string: string str = "ABCDEFD"; What is the value of str.find('D');</p>	3		
<p>Given the string: string str = "ABCDEFD"; What is the value of str.find('G');</p>	string::npos		
<p>Given: string str = "ABCDEFGHIJKLM"; What is the value of str.substr(1, 4);</p>	BCD		
<p>Given: string str = "ABC"; What is the value of str.substr(4, 5);</p>	Runtime error. Start must be < 4		
<p>Given: string str = "Gone with the wind"; What is the value of str.substr(5, 4);</p>	with		
<p>What variables are accessible at the marked location?</p> <pre>void sample(void) { int i = 3; if (i == 3) { int j; } { int i; } int k; //What is accessible? }</pre>	i, k		
<p>What prints? (Assume all includes, etc.)</p> <pre>int val = 1; while (val++ < 5) cout << val << " ";</pre>	2 3 4 5		
<p>What prints? (Assume all includes, etc.)</p> <pre>int val = 1; while (val++ < 5); cout << val << " ";</pre>	6		
<p>What prints? (Assume all includes, etc.)</p> <pre>int val = 1; while (val < 5); cout << val++ << " ";</pre>	Endless loop		
<p>What prints? (Assume all includes, etc.)</p> <pre>int val = 1; while (val < 5) cout << val++ << " ";</pre>	1 2 3 4		
<pre>int i = 5; while(--i) cout << i; What prints here?</pre>	4 3 2 1		
<pre>int i = 5; while(i) cout << i--; What prints here?</pre>	5 4 3 2 1		
<pre>int i = 5; while(i--) cout << i; What prints here?</pre>	4 3 2 1 0		



<pre>int x = 4; do { x -= 5; x++; } while (x >= 0);</pre>	
<p>What's printed?</p> <pre>string s = "happy"; int i = -1, len = s.length(); while (i < len - 1) cout << s[++i];</pre>	happy
<p>Assume x is an int with the value 4. What prints?</p> <pre>if (x <= 2) { if (x == 4) cout << "one"; } else cout << "two";</pre>	two
<p>Assume x is an int with the value 4. What prints?</p> <pre>if (x <= 2) if (x == 4) { cout << "one"; } else cout << "two";</pre>	Nothing. Runs but prints nothing.
<p>What's printed?</p> <pre>string s = "happy"; int i = -1; while (i < s.length() - 1) cout << s[++i];</pre>	nothing
Functions that do not have a return type are called ___ functions.	void
The heading of the function is also called the ___	Both B(signature) and C(interface) are correct
A variable or expression listed in a call to a function is called the ____.	Both B(actual parameter) and C(argument) are acceptable terms
<pre>int next(int x) { return (x + 1); }</pre> <p>Given this function, what would be printed by cout << next(next(5));?</p>	7
Which statement below about prototypes and function headers is true?	Prototypes end with a semicolon; headers do not.
<pre>int test(float, char);</pre> <p>Given this prototype, which of these is valid?</p>	<pre>int u = test(5.0, '*');</pre>
<pre>void one(int a, int& b) { a = 17; b = first + 1; } ... int j = 4, k = 3; one(j, k);</pre> <p>What are the values of j and k after this?</p>	4 and 18
<p>Consider this function call. g(1, 2);</p> <p>Which of these overloaded functions will be invoked by this call?</p>	<pre>float g(int value, int count);</pre>
<p>If this is legal and compiles, the call to doStuff() results in the assignment:</p> <pre>void doStuff(int parVal, int& parRef) { parVal = 100; parRef = 222; } ... int n1 = 1, n2 = 2; doStuff(n1, n2);</pre>	n2 = 222
<pre>int test(float, char = '*');</pre> <p>Given this prototype, which of these is valid?</p>	<pre>cout << test(12); cout << test(12.0, '&'); int u = test(5.0F); cout << test('12', '&');</pre>



Given this prototype and variable definition, which of these function calls are legal (that is, they will compile)?	
<pre>void f(float n); int a = 7;</pre> <p>Given this prototype and variable definition, which of these function calls are illegal (that is, they will NOT compile)?</p>	<pre>f(&a);</pre>
<pre>void f(float& n); int a = 7;</pre> <p>Given this prototype and variable definition, which of these function calls are illegal (that is, they will NOT compile)?</p>	None of these, because you must pass in a float for the actual parameter when the fnx is called.
<pre>string s = 'happy'; int n = countVowels(s);</pre> <p>What is the correct prototype for countVowels?</p>	<pre>int countVowels(const string&);</pre>
<pre>int f(int&); int f(int); int f(int, int); int a = 7;</pre> <p>Assume the overloaded functions shown here. Which of the following function calls will fail to compile?</p>	<pre>f(a);</pre>
<pre>int f(int&); int f(const int&); int f(char, char); int a = 7;</pre> <p>Assume the overloaded functions shown here. Which of the following function calls will fail to compile?</p>	None of these fail to compile.
<pre>string str = "cat"; string result = upper(str);</pre> <p>What is the correct prototype for this function?</p>	<pre>string upper(const string&);</pre>
<pre>string str = "cat"; upper(str); cout << str; // "CAT"</pre> <p>What is the correct prototype for this function?</p>	<pre>string upper(string&);</pre>
<pre>int a = 3; int ans = add(a, 5); cout << ans; // 8</pre> <p>What is the correct prototype for this function?</p>	<pre>int add(int, int);</pre>
<pre>int a; bool OK; OK = getInt('Integer? ', a);</pre> <p>What is the correct prototype for this function?</p>	<pre>bool getInt(const string&, int);</pre>
<pre>ifstream inFile("test.txt"); char ch; while (inFile.get(ch)) if (isupper(ch)) cout << toupper(ch);</pre> <p>If text.txt contains 'Who Is 24601?' What is printed?</p>	None of these
<p>If text.txt contains: If I saw an Aardvark I would scream! What is printed?</p> <pre>ifstream inFile("test.txt"); char ch; int i = 0; while (inFile.get(ch)) if (tolower(ch) == 'a') i++; cout << i << endl;</pre>	6



<p>results.txt contains lines of text like this:</p> <p>Smith 94</p> <p>Jones 75</p> <p>What is the legal way of reading a student's name and the student's scores in the "results.txt" file?</p>	
<pre>ifstream inFile; inFile.open("c:\\file.txt"); char ch; int n = 0; while (inFile.get(ch)) if (isdigit(ch)) { inFile.unget(); inFile >> n; cout << n; } If file.txt contains: Four and 20 blackbirds! What is printed?</pre>	None of these(missing \ for escaping \ in when trying to access file path)
Which of the following statements displays 123 as 00123?	cout << setw(5) << setfill('0') << 123;
Which of the following statements displays the value of x, which contains 986.2345, as 986.2?	cout << setprecision(1) << fixed << x;
Which of the following is the correct way to call the open function on an ofstream object named writestr?	writestr.open("File.txt")
What is the correct way to pass an fstream object as a parameter to a function?	As a reference parameter.
<p>Assume "scores.txt" does not exist. What is true?</p> <pre>ofstream out("scores.txt"); out << "Peter" << " " << 20 << endl; out << "John" << " " << 50 << endl;</pre>	Creates a new file scores.txt and writes data to it.
<p>What does this code do?</p> <pre>int i = 0; ifstream inFile("test.txt"); string myword; while (inFile >> myword) i++; cout << i << endl;</pre>	Counts the number of words in the file
<p>The input is: 23.57dogs.</p> <p>What value is stored in a?</p> <pre>int n; double a; cin >> n >> a;</pre>	Nothing, but no runtime error (cin fails)
<p>The input is: 23.57dogs.</p> <p>What value is stored in n?</p> <pre>int n; double a; cin >> a >> n;</pre>	Nothing, but no runtime error (cin fails)
<p>The input is: 23.57dogs.</p> <p>What value is stored in s?</p> <pre>int n; double a; string s; cin >> n >> a >> s;</pre>	Nothing, but no runtime error (cin fails)
<p>The input is: 23.57dogs.</p> <p>What value is stored in a?</p> <pre>int n; double a; string s; cin >> n >> s >> a;</pre>	Still waiting (blocked) for input
<pre>int ch; while (ch = cin.get() != EOF) cout.put(ch); This loop is supposed to print and echo all of the characters in input. What is the error?</pre>	You must have extra parentheses inside the while loop condition



<pre>cout.put(ch);</pre> <p>This loop is supposed to print and echo all of the characters in input. What is the error?</p>	
<p>Match the letter of the variable in the figure with the correct value or expression below.</p> <pre>string s{"walk the plank"}; auto a = s.find('a'); auto b = s.find('a', 3); auto c = s.find("nk"); auto d = s.find("Walk");</pre>	<pre>d : string::npos a : 1 c : 12 b : 11</pre>
<p>To produce one of two values (of any type) in an expression, use:</p>	<p>a conditional operator</p>
<p>A C++ string that contains Unicode characters should be preceded by:</p>	<p>u8</p>
<p>What value is stored in a after this runs?</p> <pre>string s{"abcdefg"}; auto a = s.substr(3);</pre>	<p>"defg"</p>
<p>Either/or decisions should use:</p>	<p>if . . . else</p>
<p>Which of these selects a character (char) from a string?</p>	<pre>auto a = s[0];</pre>
<p>The relative order of two variables is tested using:</p>	<p>a relational operator</p>
<p>Assume c is a char variable. What type is the variable a?</p> <pre>string s{"guten tag"}; auto len = s.size(); auto a = s.front(); s.at(len) = a; s[len] = c;</pre>	<p>char</p>
<p>Assume a is 13 and b is 10; what prints?</p> <pre>string s{"feed the fish"}; cout << s.substr(a, b) << endl;</pre>	<p>""</p>
<p>Leveled decisions, such as processing income taxes are best handled with:</p>	<p>if . . . if . . . else . . . else</p>
<p>The string find() member function may be used to search for a substring.</p>	<p>True</p>
<p>Data member is the term used in C++ for what is called a method in Java</p>	<p>False</p>
<p>s.back() = 'x'; changes the last character in the string object s to 'x'.</p>	<p>True</p>
<p>Calling s.at(1) returns a copy of the second character in the string object s.</p>	<p>False</p>
<p>Assume c is a char variable. Which line produces a syntax error?</p> <pre>string s{"guten tag"}; auto len = s.size(); auto a = s.front(); s.at(len) = a; s[len] = c;</pre>	<p>None of these</p>
<p>What header file do you include to call the isupper() function?</p>	<p><cctype></p>
<p>Assume c is a char variable. What value s stored in the variable a?</p> <pre>string s{"guten tag"}; auto len = s.size(); auto a = s.front(); s.at(len) = a; s[len] = c;</pre>	<p>'g'</p>
<p>To combine several test conditions to produce a single Boolean value, use:</p>	<p>a logical operator</p>
<p>What value is stored in a after this runs?</p> <pre>string s{"ABC"}; auto a = s.substr(4, 5);</pre>	<p>Runtime error because start (4) must bet 0..3</p>
<p>In Line 2, what is the request?</p> <pre>string s{"happy"}; auto pos = s.find('y');</pre>	<p>find</p>
<p>Multiple possible outputs, testing a single condition, use:</p>	<p>if . . . else . . . if . . . else</p>



<pre>[1] int n1 = 4; [2] double n2 = 3.145; [3] unsigned char n3 = 158; [4] int n4 = n2; [5] int& r1 = n1; [6] int& r2 = n2; [7] double& r3 = n1; [8] const int& r4 = n2;</pre>	
The string find() member function takes either a string or character as an argument.	True
The string find() member function throws an exception if the target cannot be found.	False
The toupper() member function ignores case when it searches.	False
<p>Match the letter of the variable in the figure with the correct statement below.</p> <pre>string s{"ahoy"}; auto a = s.size(); auto b = s.back(); auto c = s.at(0); auto d = s.substr(a); auto e = s.substr(0, 1);</pre>	<pre>[e] : "a" [c] : 'a' [a] : string::size_type [d] : "" [b] : 'y'</pre>
<p>In Line 2, what is the explicit argument?</p> <pre>string s{"happy"}; auto pos = s.find('y');</pre>	'y'
<p>Assume c is a char variable. Which line produces undefined behavior?</p> <pre>string s{"guten tag"}; auto len = s.size(); auto a = s.front(); s.at(len) = a; s[len] = c;</pre>	5
Decisions based on numbered blocks of code are best handled with:	switch
<p>What value is stored in a after this runs?</p> <pre>string s{"ABCDEFGHIIJKLM"}; auto a = s.substr(1, 4);</pre>	"BCDE"
<p>This compiles, runs and prints 4, 3. What is the correct prototype?</p> <pre>int x = 3, y = 4; swap(x, y); cout << x << ", " << y << endl;</pre>	void swap(int& a, int& b);
<p>What value is stored in a after this runs?</p> <pre>string s{"ABCDEFD"}; auto a = s.find('D');</pre>	3
<p>What type is the variable len?</p> <pre>string s{"guten tag"}; auto len = s.size(); auto a = s.front(); s.at(len) = a; s[len] = c;</pre>	string::size_type
<p>Which lines compile and return string objects?</p> <pre>[1] string s{"shiver me timbers"}; [2] auto len = s.size(); [3] s.front() = 'S'; [4] s.back() = "S"; [5] s[len] = 'X'; [6] s.substr(0, 1) = "W"; [7] auto a = s.substr(0, 100); [8] auto b = s.substr(4, 3); [9] auto c = s.substr(len);</pre>	7, 8, 9
s.at(0) = 'c'; changes the first character in the string object s to 'c'.	True
The getline() function is part of the string class.	False
Calling s.at(0) returns the same reference as s.front().	True



<pre>string s("guten tag"); auto len = s.size(); auto a = s.front(); s.at(len) = a; s[len] = c;</pre>	
<p>This compiles, runs and prints 12. What is the correct parameter declaration for x?</p>	<p>int& x</p>
<p>To enter a Unicode character into a C++ string, use an escape sequence starting with:</p>	<p>\U</p>
<p>In Line 2, what is the receiver?</p> <pre>string s{"happy"}; auto pos = s.find('y');</pre>	<p>s</p>
<p>Assume a is 5 and b is 3; what prints?</p> <pre>string s("feed the fish"); cout << s.substr(a, b) << endl;</pre>	<p>"the"</p>
<p>In Line 2, what is the parameter?</p> <pre>string s{"happy"}; auto pos = s.find('y');</pre>	<p>None of these</p>
<p>All of these are declared in the <string> header; which are member functions?</p>	<p>size() front() find() at()</p>
<p>String parameters should be passed to functions:</p>	<p>by constant reference (const string& s) when not modified in the function. by reference (string& s) when modified in the function</p>
<p>Calling s.at(1) returns a reference to the second character in the string object s.</p>	<p>True</p>
<p>What is stored in s after this code runs?</p> <pre>string s("xyzw"); s.at(2) = 'Y';</pre>	<p>xyYw</p>
<p>In C++ a char may be one, two or three bytes, when using UTF-8.</p>	<p>False</p>
<p>How many variables appear in the following code segment?</p> <pre>int n = 4; int& r1 = n; auto& r2 = r1; r1 = 3; r2 = 5; cout << n << endl;</pre>	<p>1</p>
<p>s.at(0) = "c"; changes the first character in the string object s to 'c'.</p>	<p>False</p>
<p>What value is stored in a after this runs?</p> <pre>string s("ABCDEFD"); auto a = s.find('G');</pre>	<p>string::npos</p>
<p>What does this code segment print?</p> <pre>int n = 4; int& r1 = n; auto& r2 = r1; r1 = 3; r2 = 5; cout << n << endl;</pre>	<p>5</p>
<p>Which of these lines are legal?</p> <pre>[1] int n1 = 4; [2] double n2 = 3.145; [3] unsigned char n3 = 158; [4] int n4 = n2; [5] int& r1 = n1; [6] int& r2 = n2; [7] double& r3 = n1; [8] const int& r4 = n2;</pre>	<p>4, 5, 8</p>



<pre>[1] string s{"shiver me timbers"}; [2] auto len = s.size(); [3] s.front() = 'S'; [4] s.back() = "S"; [5] s[len] = 'X'; [6] s.substr(0, 1) = "W"; [7] auto a = s.substr(0, 100); [8] auto b = s.substr(4, 3); [9] auto c = s.substr(len);</pre>	
<p>In Line 2, what is the implicit argument?</p> <pre>string s{"happy"}; auto pos = s.find('y');</pre>	the address of s
<p>Assume a is 14 and b is 10; what prints?</p> <pre>string s{"feed the fish"}; cout << s.substr(a, b) << endl;</pre>	Runtime error
<p>In Line 2, what is the result of this function call?</p> <pre>string s{"happy"}; auto pos = s.find('y');</pre>	pos
<p>Assume a is 9 and b is 10; what prints?</p> <pre>string s{"feed the fish"}; cout << s.substr(a, b) << endl;</pre>	"fish"
One-way, independent decisions use:	if
<p>Assume c is a char variable. What type is the expression s.last()?</p> <pre>string s{"guten tag"}; auto len = s.size(); auto a = s.front(); s.at(len) = a; s[len] = c;</pre>	None of these
<p>Which lines cause syntax errors?</p> <pre>[1] string s{"shiver me timbers"}; [2] auto len = s.size(); [3] s.front() = 'S'; [4] s.back() = "S"; [5] s[len] = 'X'; [6] s.substr(0, 1) = "W"; [7] auto a = s.substr(0, 100); [8] auto b = s.substr(4, 3); [9] auto c = s.substr(len);</pre>	4, 6
<p>End a block of source code</p> <p>Meaning of value returned from a function</p> <p>Required to document functions, global variables and constants</p> <p>Begin a block of source code</p> <p>Your name</p> <p>Information about the library</p> <p>When was it created?</p> <p>Name and meaning for a parameter</p>	<p>@endcode</p> <p>@return</p> <p>@file</p> <p>@code</p> <p>@author</p> <p>@version</p> <p>@date</p> <p>@param</p>
Which of these documentation tags are used in a file comment?	<p>@version</p> <p>@author</p> <p>@date</p> <p>@file</p>
Which of these documentation tags are used in a function comment?	<p>@return</p> <p>@param</p> <p>@endcode</p> <p>@code</p>
<p>What kind of error is this?</p> <pre>ex1.cpp:7:10: error: expected ';' after expression a = 4 ^ ;</pre>	Syntax error (mistake in grammar)

Total combine1		Study	
<pre>ex1.cpp:6:5: error: use of undeclared identifier 'a' a = 4; ^</pre>			
<p>What kind of error is this?</p> <pre>ex1.cpp:6:12: error: no viable conversion from 'int' to 'string' string a = 15; ^ --</pre>	Type error (wrong initialization or assignment)		
<p>What kind of error is this?</p> <pre>ex1.cpp:7:9: warning: missing terminating '"' character a = "hello world; ^ ex1.cpp:7:9: error: expected expression</pre>	Syntax error (mistake in grammar)		
<p>What is the output of the following?</p> <pre>string s = "12345"; int i = 1; while (i < 5) { cout << s.substr (i, 1); i++; }</pre>	2345		
<p>What is the output of the following?</p> <pre>string s = "abcde"; int i = 1; while (i < 5) { cout << s.substr (i, 1); i++; }</pre>	bcde		
<p>What is the output of the following?</p> <pre>int i = 1; while (i < 10) { cout << i << " "; i = i + 2; if (i == 5) { i = 9; } }</pre>	1 3 9		
<p>What is the output of the following?</p> <pre>int i = 1; while (i <= 10) { cout << "Inside the while loop" << endl; i = i * 11; }</pre>	"Inside the while loop" will be displayed only once.		
<p>What is the output of the following?</p> <pre>int i = 1; int sum = 0; while (i <= 11) { sum = sum + i; i++; } cout << "The value of sum is " << sum;</pre>	The value of sum is 66		
<p>What is the output of the following?</p> <pre>int i = 0; while (i != 11) { cout << i << " "; i = i + 2; }</pre>	0 2 4 6 8 10 12 14 (infinite loop)		
<p>What is the output of the following?</p> <pre>int i = 1; int sum = 0; while (i <= 13) { sum = sum + i; i = i + 3; } cout << "The value of sum is " << sum;</pre>	The value of sum is 35		


Total combine1		Study	
<pre>int i = 0; while (i != 15) { cout << "So far so good" << endl; i++; }</pre>			
What is the output of the following? <pre>int i = 1; while (i < 20) { cout << i << " "; i = i + 2; if (i == 15) { i = 19; } }</pre>		1 3 5 7 9 11 13 19	
What is the output of the following? <pre>int i = 0, j = 0; while (i < 125) { i = i + 2; j++; } cout << j << endl;</pre>		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 run-time error		False	
An undeclared error message is a linker error		False	
Implementation files may use the statement using namespace std;		True	
Implementation files must explicitly qualify each name from the standard library with std::		False	
Parameter names are optional in the function prototype		True	
Parameter names are optional in the function definition		False	
A tool named Doxygen is often used to generate HTML user docs from C++ code.		True	
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	
Which prototypes in the following header file contain errors? <pre>#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</pre>		f1	
Which prototypes in the following header file contain errors? <pre>#ifndef EXAMPLE_H #define EXAMPLE_H string f1(int a); int f2(double); void f3(std::string& s, int n); double f4(); #endif</pre>		f1 f3	



<pre>#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</pre>	
<p>Which of these are dependencies?</p> <p>EXE=digit-tester OBS=client.o digits.o \$(EXE): \$(OBS) \$(CXX) \$(CXXFLAGS) \$(OBS) -o \$(EXE)</p>	<p>digits.o client.o</p>
<p>Which of these are targets?</p> <p>EXE=digit-tester OBS=client.o digits.o \$(EXE): \$(OBS) \$(CXX) \$(CXXFLAGS) \$(OBS) -o \$(EXE)</p>	<p>\$(EXE) digit-tester</p>
<p>How many lines of output are printed?</p> <pre>int i = 0; while (i != 9) { cout << "Loop Execution" << endl; i++; }</pre>	<p>9</p>
<p>What is the output of the following?</p> <pre>int i = 0; while (i != 9) { cout << i << " "; i = i + 2; }</pre>	<p>0 2 4 6 8 10 12 14 (infinite loop)</p>
<p>What is the output of the following?</p> <pre>int i = 1; while (i != 9) { cout << i << " "; i++; if (i = 9) { cout << "End"; } }</pre>	<p>1 End</p>
<p>How many lines of output are printed?</p> <pre>int count = 0; while (count != 9) { cout << "Monster Mash" << endl; if ((count % 2) == 0) { count++; } else { count--; } }</pre>	<p>Infinite</p>
<p>What is the output of the following?</p> <pre>bool token = false; while (token) { cout << "Hello World!" << endl; }</pre>	<p>No output</p>



<pre>bool token1 = true; while (token1) { for (int i = 0; i < 5; i++) { cout << "Hello there" << endl; } token1 = false; }</pre>	
<p>What is the output of the following?</p> <pre>bool val1 = true; bool val2 = false; while (val1) { if (val1) { cout << "Hello" << endl; } val1 = val2; }</pre>	"Hello" will be displayed only once.
<p>Which line in the function "skeleton" below contains an error?</p> <pre>#include "digits.h" // 1. int firstDigit(int n); // 2. { // 3. return 0; // 4. } // 5.</pre>	// 2.
<p>Which line in the function "skeleton" below contains an error?</p> <pre>#include "digits.h" // 1. int firstDigit(int n) // 2. { // 3. return 0; // 4. }</pre>	None of these
<p>Which line in the function "skeleton" below contains an error?</p> <pre>#include "borgia.h" // 1. void primoTiara(int n) // 2. { // 3. return 0; // 4. } // 5.</pre>	// 4.
<p>What kind of error is this?</p> <pre>ex1.cpp:7: undefined reference to `f0'</pre>	Linker error (something is missing when linking)
<p>What kind of error is this?</p> <pre>~/workspace/ \$./ex1 The Patriots won the 2018 Super Bowl</pre>	None of these
<p>What kind of error is this?</p> <pre>terminate called after throwing an instance of 'std::out_of_range'</pre>	Runtime error (throws exception when running)
<p>What kind of error is this?</p> <pre>Segmentation fault</pre>	Operating system signal or trap
<p>In a library, the implementation file:</p>	consists of function definitions
<p>In a library, the interface file:</p>	consists of declarations or prototypes
<p>In a library, the client or test program:</p>	consists of function calls
<p>In a library, the makefile:</p>	consists of instructions that produce the executable
<p>In a client file you should compare your function's value to the _____?</p>	expected value
<p>In a client file, the value returned from calling your function is the_____?</p>	actual value
<p>Loops that do some processing and then compare the results against a boundary condition are called _____?</p>	limit loops
<p>An incomplete, yet compilable, linkable and executable function is called a _____?</p>	stub
<p>Which of these program organization schemes does not work?</p>	Call your functions and define them afterwards.
<p>Which of these may go into a header file?</p>	function prototypes constant definitions

Total combine1		Study	
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		digit-tester	
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	
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	



<pre>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); }</pre>	
<p>What prints here?</p> <pre>auto a = 3, b = 3; cout << (a != b ? "panda": "tiger") << endl;</pre>	tiger
<p>What prints here?</p> <pre>auto a = 4, b = 3; cout << (a == b ? "panda": a % 2 ? "stork": "tiger") << endl;</pre>	tiger
<p>What prints here?</p> <pre>auto a = 3, b = 3; cout << (a == b ? "panda": "tiger") << endl;</pre>	panda
<p>What prints here?</p> <pre>auto a = 3, b = 3; cout << (a != b ? "panda": a % 2 ? "stork": "tiger") << endl;</pre>	stork
<p>What prints here?</p> <pre>auto a = 3, b = 3; cout << a == b ? "panda" : "tiger" << endl;</pre>	Does not compile
<p>Function overloading allows you to write several different functions that have the same name.</p> <p>Function overloading lets you call a single function in several different ways.</p>	<p>True</p> <p>False</p>
<p>Overloaded functions have the same name but different parameter types.</p> <p>Overloaded functions have the same name but different parameter names.</p>	<p>True</p> <p>False</p>
<p>In a while loop, (condition) is followed by a semicolon.</p> <p>A while loop is a hasty or unguarded loop.</p>	<p>False</p> <p>False</p>
<p>What prints here?</p> <pre>auto a = 1; switch (a) { case 1: cout << "1"; break; case 2: cout << "2"; break; default: cout << "3"; } cout << endl;</pre>	1
<p>What prints here?</p> <pre>auto a = 2; switch (a) { case 1: cout << "1"; break; case 2: cout << "2"; break; default: cout << "3"; } cout << endl;</pre>	2
<p>What prints here?</p> <pre>auto a = '1'; switch (a) { case 1: cout << "1"; break; case 2: cout << "2"; break; default: cout << "3"; } cout << endl;</pre>	3



<pre>auto a = 1; switch (a) { case 1: cout << "1"; case 2: cout << "2"; } cout << endl;</pre>	
<p>What prints here?</p> <pre>auto a = 1; switch (a) { case 1: cout << "1"; case 2: cout << "2"; case 3: } cout << endl;</pre>	Does not compile
<p>What prints here?</p> <pre>double a = 1; switch (a) { case 1: cout << "1"; case 2: cout << "2"; } cout << endl;</pre>	Undefined behavior
<p>What prints here?</p> <pre>auto a = 'A'; switch (a) { case 64: cout << "?"; case 65: cout << "A"; case 66: cout << "B"; } cout << endl;</pre>	A But should be AB
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.	True
Default arguments allow you to write several different functions that have the same name.	False
Default arguments may only be used with value parameters.	True
Default arguments may only be used with reference parameters.	False
Default arguments may be used with both value and reference parameters.	False
Default arguments appear only in the function prototype.	True
Default arguments appear only in the function implementation.	False
Fatal error messages should be printed to cerr.	True
Fatal error messages should be printed to cout.	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



<pre>void f(... str); int main() { string s = "hello"; f(s); }</pre>	
<p>To allow f() to accept the argument passed here, the parameter str should be declared as:</p> <pre>void f(... str); int main() { f("hello"); }</pre>	const string&
<p>To allow f() to change the argument passed here, the parameter str should be declared as:</p> <pre>void f(... str); int main() { f("hello"); }</pre>	It is not possible for f() to change the argument passed here.
<p>A function where an argument is converted to match a parameter</p> <p>When more than one match is found for the proffered arguments.</p> <p>A function where each argument is the same type as the corresponding parameter.</p> <p>A group of functions with the same name.</p> <p>A group of functions that have the same name and the correct number of parameters.</p> <p>When no match is found for the proffered arguments</p>	<p>best match</p> <p>ambiguity</p> <p>exact matches</p> <p>candidate set</p> <p>viable set</p> <p>empty set</p>
<p>Examine the following variables and function calls</p> <p>Match each item with the correct statement below.</p> <pre>int able = 3; int baker = f1(able); cout << able << baker << endl; // 64</pre> <p>int charlie;</p> <pre>f2("hello", charlie); cout << charlie << endl; // Hello Carl</pre>	<p>Returned value --> baker</p> <p>Output argument (parameter) --> Charlie</p> <p>Input argument (parameter) --> Hello</p> <p>Input/output argument (parameter) --> able</p>
Which of these are not ways that functions may be overloaded?	<p>different function name</p> <p>different return type</p> <p>different parameter names</p>
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
<p>Given the overloaded functions prototypes and the variable definition below, which of the function calls will fail to compile?</p> <pre>int f(int&); int f(int); int f(int, int); int a = 7;</pre>	f(a);
<p>Given the overloaded functions prototypes and the variable definition below, which of the function calls will fail to compile?</p> <pre>int f(int&); int f(const int&); int f(int, int); int a = 7;</pre>	None of these fail to compile

Total combine1		Study	
<pre>int i = 1; int n; cin >> n; do { i++; cin >> n; } while (n % 2); cout << i << endl;</pre>			
<p>Assume that the input is 5 5 4 3 5. What will print?</p> <pre>int i = 1; int n; do { cin >> n; i++; } while (n % 2); cout << i << endl;</pre>	4		
<pre>int i = 1; int n; do { cin >> n; i++; } while (n % 2); cout << i << endl;</pre>	lvalues		
<p>Examine this code. Which is the best prototype?</p> <pre>int age; string name = read("Enter your name, age: ", age);</pre>	<code>string read(const string&, int&)</code>		
<p>What prints?</p> <pre>string str = "Hello"; for (int i = str.size() - 1; i >= 0; i--) cout << str.at(i);</pre>	olleH		
<p>What prints?</p> <pre>string str = "Hello"; for (size_t i = str.size() - 1; i >= 0; i--) cout << str.at(i);</pre>	Crashes when run		
<p>What prints?</p> <pre>string str = "Hello"; for (auto i = 0, len = str.size(); i < len; i++) cout << str.at(i);</pre>	Does not compile		
<p>Which of these prototypes is the best one to use in this circumstance?</p> <pre>int main() { string str{"To be or not to be."}; cout << "Most common letter is " << mostCommon(str) << endl; }</pre>	<code>char mostCommon(const string&);</code>		
<p>Which of these prototypes is the best one to use in this circumstance?</p> <pre>int main() { string str{"TO BE OR NOT TO BE"}; properCase(str); cout << str << endl; }</pre>	<code>void properCase(string&);</code>		
<p>Examine this code. Which is the best prototype?</p> <pre>string s = "dog"; cout << upper(s) << endl; // DOG cout << s << endl; // dog</pre>	<code>string upper(const string&)</code>		
<p>Examine this code. Which is the best prototype?</p> <pre>string s = "dog"; upper(s); cout << s << endl; // DOG</pre>	<code>string upper(const string&)</code>		
Arguments passed to a function that has a non-constant reference parameter must be:	lvalues		



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
<div>What prints here?</div> <div>int i = 5; while (--i) cout << i; cout << endl;</div>	4321
<div>What prints here?</div> <div>int i = 5; while (i--) cout << i; cout << endl;</div>	43210
<div>What prints here?</div> <div>int i = 5; while (i) cout << --i; cout << endl;</div>	43210
<div>What prints here?</div> <div>int i = 5; while (i) cout << i--; cout << endl;</div>	54321
<div>What prints here?</div> <div>int i = 5; while (i); cout << i--; cout << endl;</div>	Infinite loop
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);
<div>Complete the following code in the echo filter program.</div> <div>char ch; while (cin.get(ch)) _____;</div>	cout.put(ch)
<div>Complete the following code in the lower filter program.</div> <div>char ch; while (cin.get(ch)) cout.put(_____);</div>	tolower(ch)
<div>Complete the following code in the upper filter program.</div> <div>char ch; while (cin.get(ch)) cout.put(_____);</div>	toupper(ch)
<div>Complete the following code in the echo filter program.</div> <div>char ch; while (_____) cout.put(ch);</div>	cin.get(ch)
<div>Assume the user types "brown cow" when this code runs. What type is ch2?</div> <div>char ch1; auto ch2 = cin.get(ch1);</div>	istream&
<div>Assume the user types "brown cow" when this code runs. What prints?</div> <div>int n; if (cin >> n) cout << "X\n"; else cout << "Y\n";</div>	Y

Total combine1		Study	<div><div></div><div></div><div></div></div>
char ch1; auto ch2 = cin.get(ch1);			
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 a reference to the input stream The expression cin.get(ch) returns the next character from input	True 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		

Total combine1		Study	<div>100%</div>
Assume you have a char variable named ch. How do you look ahead before reading a character?		<div>cin.get(ch); cin.unget(ch); cin.putback(ch); cin.seek(ch); cin.peek(ch);</div>	
2 Q U E S T I O N S		-- > None of these	
Which line runs the dwk program and gets its input from a file named y.data?		./dwm < 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		X	
Discard both output and errors		rm x > /dev/null/2>&1	
Write output to a new file named z		X	
Read the input from the file named z		cat < z	
Write errors to a new file named z		cat x 2>z	
Send the output to the input of the program named 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	
Has a single char& parameter		get()	
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	



<pre>ifstream in("temp.txt"); char c; while (in.get(c)) { if (isupper(c)) cout << toupper(c); }</pre>	
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; outopen("outtxt");
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? <pre>ifstream in("temp.txt"); char c; int i = 0; while (in.get(c)) { if (tolower(c) == 'a') i++; } cout << i << endl;</pre>	6
The return value of the getline() function is an input stream object	True
The return value of the getline() function is a string object.	False
When writing a function with stream parameters, always use the most general type of stream that meets the specification	True
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.	True
The cout object is an instance of the ofstream class	False
A loop that reads data until the input stream signals that it is done is called a data loop	True
A loop that reads data until the input stream signals that it is done is called a sentinel loop	False
In the primed loop pattern, you read data before the loop and at the end of the loop.	True
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

Total combine1		Study	<div></div>
The getline() function is a non-member function in the string library		False	
The getline() function is a member function in the string class			
The getline() function is a member function in the istream class.			
		False	
		True	
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			
		False	
To use a disk file as a data stream source or sink, use the <ofstream> header			
		False	
		True	
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			
		False	
		True	
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			
		False	
		True	
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			
		False	
		True	
The cin object is an instance of the istream class			
The cin object is an instance of the ifstream class			
		False	
		True	
Stream parameters should always be passed to functions by reference			
Stream parameters should always be passed to functions by const reference			
		False	
		True	
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.			
		False	
In the flag-controlled-pattern, you read data before the loop and at the end of the loop			
		False	
		True	
In the loop-and-a-half, you use a break statement to exit the loop when the sentinel is found			
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	
		True	
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			
		False	
If an output stream's file is missing when you try to open it, its fail() member function returns false.			
		True	
		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			
		False	
		True	
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			
		False	



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
Read and write characters to memory using streams	sstream
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

<pre>int upper = 0; char ch; while (in.get(ch)) { if (ch >= 'A' && ch <= 'Z') upper++; }</pre>	
<p>Which of the following loop patterns are used here?</p> <pre>int n; in >> n; while (abs(n)) { out << n % 4 << endl; n /= 4; }</pre>	limit loop
<p>Which of the following loop patterns are used here?</p> <pre>auto len = str.size(); while (len) out << str.at(--len);</pre>	counter-controlled loop
<p>Which of the following loop patterns are used here?</p> <pre>string s("hello CS 150"); for (auto e : s) { if (toupper(e)) out.put('x'); }</pre>	iterator or range loop
<p>Which of the following loop patterns are used here?</p> <pre>string s("hello CS 150"); for (auto e : s) { if (toupper(e)) break; }</pre>	iterator or range loop loop-and-a-half
<p>Which of the following loop patterns are used here?</p> <pre>string s("Hello CS 150"); while (s.size()) { if (s.at(0) == 'C') break; s = s.substr(1); } cout << s << endl;</pre>	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 /}
<p>This loop:</p> <pre>char c; while (in.get(c)) { cout << c << endl; }</pre>	illustrates raw character I/O
<p>This loop:</p> <pre>char c; while (c = in.get()) { cout << c << endl; }</pre>	illustrates line-based stream processing
<p>This loop:</p> <pre>string str; while (getline(in, str)) { cout << str << endl; }</pre>	illustrates line-based stream processing
<p>This loop:</p> <pre>string str; while (in >> str) { cout << str << endl; }</pre>	illustrates token-based stream processing

Total combine1		Study	<div>1/1</div>
<div>Smith 94</div> <div>Jones 75</div> <div>...</div> <div>Each line of text contains the student's name (a single word) and an integer score.</div> <div>What is the legal way of reading one student's information, given the following code?</div> <div>string name;</div> <div>int score;</div> <div>ifstream in("grades.txt");</div>			
<div>The file expenses.txt contains the line: Hotel, 3 nights. \$ 1,750.25. What prints?</div> <div>ifstream in("expenses.txt");</div> <div>char c;</div> <div>while (in.get(c))</div> <div>{</div> <div>if (isdigit(c)) {</div> <div>in.unget();</div> <div>double n;</div> <div>in >> n;</div> <div>cout << n << 'x';</div> <div>}</div> <div>}</div>	3x1x750.25x		
<div>The file expenses.txt contains the line: Hotel, 3 nights. \$ 1,750.25. What prints?</div> <div>ifstream in("expenses.txt");</div> <div>char c;</div> <div>while (in.get(c))</div> <div>{</div> <div>if (isdigit(c)) {</div> <div>in.unget();</div> <div>int n;</div> <div>in >> n;</div> <div>cout << n << 'x';</div> <div>}</div> <div>}</div>	3x1x750x25x		
<div>Assume that the file scores.txt does not exist. What happens?</div> <div>ofstream out("scores.txt");</div> <div>out << "Peter" << " " << 20 << endl;</div> <div>out << "John" << " " << 50 << endl;</div>	Creates a new file, scores.txt and writes two lines of text		
<div>Which line represents the necessary bounds in this loop?</div> <div>1. string s("Hello CS 150");</div> <div>2. while (s.size())</div> <div>3. {</div> <div>4. if (s.at(0) == 'C') break;</div> <div>5. s = s.substr(1);</div> <div>6. }</div> <div>7. cout << s << endl;</div>	2		
<div>Which line represents the intentional bounds in this loop?</div> <div>1. string s("Hello CS 150");</div> <div>2. while (s.size())</div> <div>3. {</div> <div>4. if (s.at(0) == 'C') break;</div> <div>5. s = s.substr(1);</div> <div>6. }</div> <div>7. cout << s << endl;</div>	4		
<div>Which line advances the loop?</div> <div>1. string s("Hello CS 150");</div> <div>2. while (s.size())</div> <div>3. {</div> <div>4. if (s.at(0) == 'C') break;</div> <div>5. s = s.substr(1);</div> <div>6. }</div> <div>7. cout << s << endl;</div>	5		
<div>What header file to you need to include to use the standard C++ error-handling classes?</div>	<stdexcept>		
<div>The logic_error and runtime_error classes are defined in the header file ____.</div>	stdexcept		

Total combine1		Study	
<pre>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"; }</pre>			
<p>What prints?</p> <pre>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"; }</pre>		one	
<p>What prints?</p> <pre>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"; }</pre> <p>➤ I F (s.size() > 2) && throw s.size() && throw s.back()</p>		five	
<p>What prints?</p> <pre>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"; }</pre> <p>➤ I F (s.size() > 5) && throw s.size() && throw s.back()</p>		two	
<p>What prints?</p> <pre>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"; }</pre> <p>➤ I F (s.size() > 2) && throw 42; && throw "goodbye";</p>		two	



<pre>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"; }</pre> <p>➤ I F (s.size() > 20) && throw 42; && (islower(s.back())) throw "goodbye";</p>	<div></div>
<p>What prints?</p> <pre>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"; }</pre> <p>➤ I F (s.size() > 2) && throw 42; && (isupper(s.back())) throw "goodbye";</p>	<p>four</p> <div></div>
<p>What is correct for # 1?</p> <pre>int main() { //1 { string s = "hello"; cout << s.at(5) << endl; } // 2 // 3 (e) { cout << e. () << endl; // 4 } }</pre>	<p>try</p> <div></div>
<p>What is correct for # 2?</p> <pre>int main() { //1 { string s = "hello"; cout << s.at(5) << endl; } // 2 // 3 (e) { cout << e. () << endl; // 4 } }</pre>	<p>catch</p> <div></div>
<p>What is correct for # 3?</p> <pre>int main() { //1 { string s = "hello"; cout << s.at(5) << endl; } // 2 // 3 (e) { cout << e. () << endl; // 4 } }</pre>	<p>exception&</p> <div></div>



<pre> { //1 { string s = "hello"; cout << s.at(5) << endl; } // 2 // 3 (e) { cout << e. () << endl; // 4 } }</pre>	
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
Which of the following blocks is designed to catch any type of exception?	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 __cplusplus 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

Total combine1		Study
The preprocessor operates on code after it has been compiled.	False	
The directives #if defined(symbol) and #ifdef symbol mean, essentially, the same thing	True	
The directives #if defined(symbol) and #ifndef symbol mean, essentially, the same thing.	False	
A catch block may handle exception classes, as well as errors where int or string are thrown	True	
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	
The heading of a try block can contain ellipses in place of a parameter	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	





<pre>cout << stoi("one") << endl;</pre>	
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.
<p>What happens when this code fragment runs?</p> <pre>istringstream in("12.5"); int n; in >> n;</pre>	n is set to 12
<p>What happens when this code fragment runs?</p> <pre>istringstream in("12"); int n; in >> n;</pre>	n is set to 12
<p>What happens when this code fragment runs?</p> <pre>istringstream in(".5"); int n; in >> n;</pre>	It sets an error state in in.
<p>What happens when this code fragment runs in C++ 11?</p> <pre>istringstream in("one"); int n; in >> n;</pre>	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
<p>Which line fails to work correctly?</p> <pre>template <typename T> void print(const T& item) { cout << item << endl; }</pre>	ANSWER --> None of these <pre>print(2 + 2); print(string("goodbye")); print(3 + 2.2); print("hello");</pre>
<p>Assume s1 and s2 are C++ string objects. Which of these calls is illegal?</p> <pre>template <typename T> void addem(T a, T b) { cout << a << " + " << b << "->" << (a + b) << endl; }</pre>	addem(1.5, 2);
<p>Which call below produces 5?</p> <pre>template <typename T> void addem(T a, T b) { cout << a << " + " << b << "->" << (a + b) << endl; }</pre>	addem<int>(3, 2.5);
<p>Assume s1 and s2 are C++ string objects. Which of these calls is illegal?</p> <pre>template <typename T> void addem(T a, U b) { cout << a << " + " << b << "->" << (a + b) << endl; }</pre>	ANSWER --> None of these <pre>addem(1.5, 2); addem(s1, s2); addem(3, 4) addem(4.5, 5.5);</pre>
<p>What happens when this code fragment compiles and runs?</p> <pre>#define N #ifdef N cout << "Hello"; #else cout << "Goodbye"; #endif</pre>	prints "Hello"
<p>What happens when this code fragment compiles and runs?</p> <pre>#define N #ifndef N cout << "Hello"; #else cout << "Goodbye"; #endif</pre>	prints "Goodbye"

Total combine1		Study	
<pre>#if __APPLE__ istringstream in(" .75"); int n = 3; in >> n; #endif</pre>			
<p>Complete the code fragment below, which is designed to throw an illegal_length exception if string variable accountNumber has more than seven characters.</p> <pre>if (accountNumber.size() > 7) { _____; }</pre>	<pre>throw illegal_length("Account number exceeds maximum length");</pre>		
<p>Examine the following code (which is legal). What is the correct prototype for an aggregate output operator?</p> <pre>struct Time { int hours{0}, minutes{0}, seconds{0}; };</pre>	<pre>ostream& operator<<(ostream& out, const Time& m);</pre>		
<p>Examine the following code (which is legal). What is the correct prototype for an aggregate output operator?</p> <pre>struct Money { int dollars{0}, cents{0}; } m1, m2;</pre>	<pre>ostream& operator<<(ostream& out, const Money& m);</pre>		
<p>Examine the following code (which is legal). Which statement is illegal?</p> <pre>struct Money { int dollars{0}, cents{0}; } m1, m2;</pre>	<pre>cout << m1 << endl;</pre>		
<p>Examine the following code (which is legal). Which statement is legal?</p> <pre>struct Money { int dollars{0}, cents{0}; } m1, m2;</pre>	<pre>m1 = m2;</pre>		
<p>Examine the following code (which is legal). Which statement is correct?</p> <pre>struct Rectangle { int length, width; };</pre>	<pre>Rectangle r;</pre>		
<p>The following is legal. Which is the correct way to access a data member in the Rectangle variable named r?</p> <pre>struct Rectangle { int length, width; };</pre>	<pre>r.length</pre>		
<p>The structure and variable definitions are fine. Which statements are legal?</p> <pre>struct Rectangle { int length, width; } big, small;</pre>	<pre>if (big.length == small.width) . . .</pre>		
<p>The following is legal. Which changes the length data member inside the variable big?</p> <pre>struct Rectangle { int length, width; } big, little;</pre>	<pre>big.length = 10;</pre>		
<p>Examine the following code (which is legal). What changes are necessary to allow the statement if (m1 == m2) ... to compile?</p> <pre>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;</pre>	<p>The name of equals() must be changed to operator==</p>		
<p>Examine the following code (which is legal). What changes are necessary to allow the statement if (m1 != m2) ... to compile?</p> <pre>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; }</pre>	<p>You must write a function named operator!=</p>		
<p>Examine the following definition. What is the syntax error?</p> <pre>struct Employee { long empID; std::string lastName; double salary; }</pre>	<p>missing a semicolon after the structure definition</p>		
<p>Examine the following definition. empID is a _____.</p> <pre>struct Employee { long empID; std::string lastName; double salary; };</pre>	<p>data member</p>		



<pre>struct Employee { long empID; std::string lastName; double salary; };</pre>	
<p>Given the following structure and variable definitions, which data members are uninitialized?</p> <pre>struct Employee { long empID{0}; std::string lastName; double salary{0}; int age = 0; }; Employee bob;</pre>	None of them (compiles)
<p>Given the following structure and variable definitions, which data members are uninitialized?</p> <pre>struct Employee { long empID; std::string lastName; double salary; int age; }; Employee bob;</pre>	salary age empID
<p>Given the following structure and variable definitions, which data members are initialized?</p> <pre>struct Employee { long empID; std::string lastName; double salary; int age; }; Employee bob;</pre>	lastName
<p>Given the following structure and variable definitions, which data members are initialized?</p> <pre>struct Employee { long empID; std::string lastName; double salary; int age; }; Employee bob{};</pre>	salary age lastName empID
<p>Given the following structure and variable definitions, which data members are default initialized?</p> <pre>struct Employee { long empID; std::string lastName; double salary; int age; }; Employee bob{777, "Zimmerman"};</pre>	age salary
<p>Given the following structure and variable definitions, which data members are default initialized?</p> <pre>struct Employee { long empID; std::string lastName; double salary; int age; }; Employee bob{777, "Zimmerman", 5000000.0, 76};</pre>	None of these



<pre>struct Money { int dollars{0}; int cents{1}; }; Money payment;</pre>	
<p>Given the following structure and variable definitions which statements are illegal?</p> <pre>struct Money { int dollars{0}; int cents{1}; }; Money payment;</pre>	<pre>payment{1} = 5; cout << Money.dollars; Money{1} = Money{0};</pre>
<p>The structure and variable definitions are fine. Which statements are legal?</p> <pre>struct R { int a, b; } a, b; struct Q { int a, b; } c, d;</pre>	<pre>c = d;</pre>
<p>YOU DONOT NEED TO REVIEW FOR TRUE / FALSE</p> 	<p>YOU DONOT NEED TO REVIEW FOR TRUE / FALSE</p> 
<p>This is the correct syntax for a C++ scoped enumeration.</p> <pre>enum class WEEKEND {SUNDAY, SATURDAY=6};</pre> <p>Structures are heterogeneous data types.</p> <p>The built-in primitive data types such as int, char and double are scalar data types.</p> <p>User-defined scalar types are created with the enum class keywords in C++.</p> <p>User-defined types that contain a single value are called scalar types.</p> <p>The standard library types such as string and vector are structured data types.</p> <p>You may create a structure variable as part of a structure definition.</p> <p>The following is an anonymous structure.</p> <pre>struct {int hours, seconds; } MIDNIGHT{0, 0};</pre> <p>Structure variables should be passed to functions by reference.</p> <p>When passing a structure variable to a function, use non-const reference if the intent is to modify the actual argument.</p> <p>The following code is legal.</p> <pre>struct {int hours, seconds; } MIDNIGHT{0, 0};</pre> <p>User-defined types that combine multiple values into a single type are called structured types.</p> <p>A structure member may be a variable of a different structure type.</p> <p>In C++, objects have value semantics; object variables contain the data members.</p> <p>Structures data members may each have a different type.</p> <p>C++ has two ways to represent records, the class and the struct.</p> <p>This is the correct syntax for a C++ scoped enumeration.</p> <pre>enum class WEEKEND {SATURDAY, SUNDAY};</pre> <p>It is illegal to include the same struct definition multiple times, even if the definitions are exactly the same.</p> <p>When passing a structure variable to a function, use const reference if the function should not modify the actual argument.</p> <p>In Computer Science, a collection of variables that have distinct names and types is called a record.</p> <p>This is the correct syntax for a C++ plain enumeration.</p> <pre>enum WEEKEND {SATURDAY, SUNDAY};</pre>	<p>All are True</p>



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};

The following definition:

vector<double> data;

creates a vector of size 0

The following definition:

vector<double> v{3, 5};

creates a vector of [3.0, 5.0]

The following definition:

vector<double> v(3, 5);

creates a vector of [5.0, 5.0, 5.0]

What prints?

vector<int> v{1, 2, 3, 4, 5};
cout << v.pop_back() << endl;

Nothing; compile-time error

What prints?

vector<int> v{1, 2, 3, 4, 5};
v.pop_back();
cout << v.front() << endl;

1

What prints?

vector<int> v{1, 2, 3, 4, 5};
v.pop_back();
cout << v.back() << endl;

4

What prints?

```
void f(vector<int> v)
{
    v.at(0) = 42;
}
int main()
{
    vector<int> x{1, 2, 3};
    f(x);
    cout << x.at(0) << endl;
}
```

1




<pre>{ v.at(0) = 42; } int main() { vector<int> x{1, 2, 3}; f(x); cout << x.at(0) << endl; }</pre>	
<p>What prints?</p> <pre>void f(const vector<int>& v) { v.at(0) = 42; } int main() { vector<int> x{1, 2, 3}; f(x); cout << x.at(0) << endl; }</pre>	Nothing; compile-time error.
<p>What does this code do?</p> <pre>int x = 0; vector<int> v{1, 3, 2}; for (auto e : v) e += x; cout << x << endl;</pre>	prints 0
<p>What does this code do?</p> <pre>int x = 0; vector<int> v{1, 3, 2}; for (auto e : v) x = e; cout << x << endl;</pre>	Finds the last element in v Prints 2
<p>What does this code do?</p> <pre>int x = 0; vector<int> v{1, 3, 2}; for (auto e : v) x += e; cout << x << endl;</pre>	Sums the elements in v Prints 6
<p>What is stored in data after this runs?</p> <pre>vector<int> data{1, 2, 3}; data.pop_back();</pre>	None of these
<p>What is the size of data, after this runs?</p> <pre>vector<int> data; data.push_back(3);</pre>	1
<p>What is stored in data after this runs?</p> <pre>vector<int> data{1, 2, 3}; data.erase(v.begin());</pre>	[2, 3]
<p>What is stored in data after this runs?</p> <pre>vector<int> data{1, 2, 3}; data.front();</pre>	[1, 2, 3]
<p>What is stored in data after this runs?</p> <pre>vector<int> data{1, 2, 3}; data.back();</pre>	[1, 2, 3]
<p>What is stored in data after this runs?</p> <pre>vector<int> data{1, 2, 3}; data.clear();</pre>	[]
<p>What is stored in data after this runs?</p> <pre>vector<int> data{1, 2, 3}; data.push_back(0);</pre>	[1, 2, 3, 0]
<p>What is stored in data after this runs?</p> <pre>vector<int> data{1, 2, 3}; data.pop_back(0);</pre>	None of these

Total combine1		Study	<div>1/1</div>
<pre>int main() { vector<int> v{1, 2, 3}; for (const auto& e : v) e = 0; cout << v.at(0) << endl; }</pre>			
<p>Which of these are true?</p> <pre>int main() { vector<int> v{1, 2, 3}; for (auto i = v.size() - 1; i >= 0; i--) // out of range for >= cout << v.at(i) << " "; cout << endl; }</pre>	<p>Crashes when run</p> <p>Prints 3 2 1</p> <p>Issues a compiler warning, but no error</p>		
<p>Which of these are true?</p> <pre>int main() { vector<int> v{1, 2, 3}; for (auto& e : v) e = 0; cout << v.at(0) << endl; }</pre>	<p>Prints 0</p>		
<p>Which of these are true?</p> <pre>int main() { vector<int> v{1, 2, 3}; for (auto e : v) e = 0; cout << v.at(0) << endl; }</pre>	<p>Prints 1</p> <p>Code runs but has no effect on v</p>		
<p>Which of these are true?</p> <pre>int main() { vector<int> v{1, 2, 3}; for (auto i = v.size() - 1; i >= 0; i--) cout << v[i] << " "; cout << endl; }</pre>	<p>Endless loop (will likely crash, but not necessarily)</p> <p>Issues a compiler warning, but no error</p> <p>Prints 3 2 1</p>		
<p>Which of these are true?</p> <pre>int main() { vector<int> v{1, 2, 3}; for (auto i = v.size(); i > 0; i--) cout << v.at(i) << " "; cout << endl; }</pre>	<p>crashes when runs</p>		
<p>Which line of code can be added to print the value 4?</p> <pre>int main() { struct S {int a, b; }; vector<S> v; S s{3, 4}; v.push_back(s); // Add code here }</pre>	<p>cout << v.at(0).b << endl;</p>		
<p>Assume vector<double> speed(5); Which line throws a run-time error?</p>	<p>ANSWER --> None of these</p> <p>cout << speed[speed.size()];</p> <p>speed[0] = speed.back()</p> <p>speed.front() = 12;</p> <p>speed.erase(speed.begin());</p>		
<p>Which defines a vector to store the salaries of ten employees?</p>	<p>vector<double> salaries(10);</p>		
<p>The following code is logically correct. What is the semantically correct prototype for mystery()?</p> <pre>vector<double> v; mystery(v);</pre>	<p>void mystery(vector<int>&);</p>		
<p>The following code is logically correct. What is the semantically correct prototype for mystery()?</p> <pre>vector<double> v{1, 2, 3}; mystery(v);</pre>	<p>Either mystery(const vector<int>&); or mystery(vector<int>&); could be correct.</p>		

Total combine1		Study	<div></div>
<pre>int main() { vector<int> v{1, 2, 3}; auto size = v.size(); cout << v.back() << endl; // 1. cout << v.front() << endl; // 2. cout << v.at(0) << endl; // 3. cout << v.at(size) << endl; // 4. cout << v.pop_back() << endl; // 5. }</pre>			
<p>Which line prints 3?</p> <pre>int main() { vector<int> v{1, 2, 3}; auto size = v.size(); cout << v.back() << endl; // 1. cout << v.front() << endl; // 2. cout << v.at(0) << endl; // 3. cout << v.at(size) << endl; // 4. cout << v.pop_back() << endl; // 5. }</pre>		1	
<p>Which statement is false? The elements in a vector:</p>		ANSWER → None of these	
		Are accessed by using an index or subscript	
		Each use the same amount of memory	
		Are are all of the same type	
		Are homogeneous	
<p>Which line compiles, but crashes when run?</p> <pre>int main() { vector<int> v{1, 2, 3}; auto size = v.size(); cout << v.back() << endl; // 1. cout << v.front() << endl; // 2. cout << v.at(0) << endl; // 3. cout << v.at(size) << endl; // 4. cout << v.pop_back() << endl; // 5. }</pre>		4	
<p>Which lines have an identical effect?</p> <pre>int main() { vector<int> v{1, 2, 3}; auto size = v.size(); cout << v.back() << endl; // 1. cout << v.front() << endl; // 2. cout << v.at(0) << endl; // 3. cout << v.at(size) << endl; // 4. cout << v.pop_back() << endl; // 5. }</pre>		2 and 3	
<p>In C++ the parameterized collection classes are called _____?</p>		templates	
<p>Classes that contain objects as elements are called?</p>		collections	
<p>Assume <code>vector<double> speed(5);</code> Which line throws a runtime error?</p>		None of these <code>speed.erase(speed.begin());</code> <code>speed.front() = 12;</code> <code>speed[0] = speed.back()</code> ANSWER → <code>cout << speed.at(speed.size());</code>	
<code>vector<int> v;</code>		Creates the empty vector []	
<code>vector<int> v(1);</code>		Creates the vector [0]	
<code>v.begin()</code>		Points to the first element in v	
<code>v.back();</code>		Returns a reference to the last element in v	
<code>v.erase(v.begin());</code>		Removes the first element in v and shifts the rest to the left	
<code>v.pop_back()</code>		Removes the last element in v	
<code>v[3];</code>		Returns a reference to the fourth element in v with no range checking	
<code>vector<int>v(2,3);</code>		Creates the vector [3,3]	



<code>v.push_back(3);</code>	Adds a new element to the end of v
<code>v.at(3);</code>	Safely returns a reference to the fourth element in v
<p>You can create vector objects to store any type of data, but each element in the vector must be the same type.</p> <p>Assume <code>vector<int> v</code>; Writing <code>cout << v.front()</code>; throws a runtime exception.</p> <p>Assume the vector v contains [1, 2, 3]. <code>v.erase(v.begin() + 2)</code>; changes v to [1, 2].</p> <p>The declaration: <code>vector<string> v(5, "bob");</code> creates a vector containing five string objects, each containing "bob".</p> <p>In the declaration: <code>vector<int> v</code>; the word <code>int</code> represents the object's base type.</p> <p>The elements of a vector are allocated contiguously.</p> <p>vector subscripts begin at 0 and go up to the vector size - 1</p> <p>The <code>clear()</code> member function removes all the elements from a vector.</p> <p>The statement <code>v.insert(v.end() + 1, 3)</code> is undefined because <code>end() + 1</code> points past the last element in the vector.</p> <p>The statement <code>v.insert(v.end(), 3)</code> appends the element 3 to the end of the vector v.</p> <p>Contiguous allocation means that the elements are stored next to each other in memory.</p> <p>The <code>push_back</code> member function adds elements to the end of a vector.</p> <p>Assume the vector v contains [1, 2, 3]. <code>v.erase(v.begin())</code>; changes v to [2, 3].</p> <p>The declaration: <code>vector<int> v(10)</code>; creates a vector object containing ten elements initialized to 0.</p> <p>Assume the vector v contains [1, 2, 3]. <code>v.pop_back()</code>; changes v to [1, 2].</p> <p>The term for classes with a base-type specification are parameterized classes.</p> <p>Assume that v contains [1, 2, 3]. The result of writing <code>cout << v[4]</code>; is undefined.</p> <p>The C++ term for classes like vector are template classes.</p> <p>A vector subscript represents the element's offset from the beginning of the vector.</p> <p>The declaration: <code>vector<string> v{"bill", "bob", "sally"}</code>; creates a vector containing three string objects.</p> <p>The declaration: <code>vector<int> v(10, 5)</code>; creates a vector object containing ten integers.</p> <p>Assuming that <code>Star</code> is a structure, the declaration: <code>vector<Star> stars(3)</code>; creates three default initialized <code>Star</code> objects.</p> <p>The declaration: <code>vector<string> v(5)</code>; creates a vector containing five empty string objects.</p> <p>Assume the vector v contains [1, 2, 3]. <code>v.erase(0)</code>; is a syntax error.</p> <p>The declaration: <code>vector<int> v</code>; creates a vector object with no elements.</p> <p>A vector represents a linear homogeneous collection of data.</p> <p>Assume <code>vector<double> v</code>; Writing <code>cout << v.back()</code>; is undefined.</p> <p>Elements in a vector are accessed using a subscript.</p> <p>Assume that v contains [1, 2, 3]. The result of writing <code>cout << v.at(4)</code>; throws a runtime exception.</p> <p>The statement <code>v.insert(v.begin(), 3)</code> inserts the element 3 into the vector v, shifting the existing elements to the right.</p>	<p>True</p>

Total combine1		Study									
<p>Vector subscripts begin at 1 and go up to the vector size.</p> <p>The statement <code>v.insert(v.end(), 3)</code> is undefined because <code>end()</code> points past the last element in the vector.</p> <p>Assume that <code>v</code> contains <code>[1, 2, 3]</code>. The result of writing <code>cout << v.at(4)</code>; is undefined.</p> <p>The C++ term for classes like vector are generic classes.</p> <p>The statement <code>v.insert(v.begin(), 3)</code> inserts the element 3 into the vector <code>v</code>, overwriting the exiting element at index 0.</p> <p>The <code>push_back</code> member function adds elements to the end of a vector as long as there is room for the elements.</p> <p>The declaration: <code>vector<int> v(10)</code>; creates a vector object containing uninitialized elements.</p> <p>The declaration: <code>vector<int> v(10, 5)</code>; creates a vector object containing five integers.</p> <p>The declaration: <code>vector<string> v(5)</code>; creates a vector containing five null pointers.</p> <p>In the declaration: <code>vector<int> v</code>; the word vector represents the object's base type.</p> <p>The declaration: <code>vector<int> v</code>; creates a vector variable but no vector object.</p> <p>Assume that <code>v</code> contains <code>[1, 2, 3]</code>. The result of writing <code>cout << v.at(4)</code>; is a compiler error.</p> <p>Vector subscripts begin at 1 and go up to the vector size.</p> <p>A vector consists of named members.</p> <p>The declaration: <code>vector<int> v(10, 5)</code>; is illegal.</p> <p>Assume <code>vector<double> v</code>; Writing <code>cout << v.back()</code>; throws a runtime exception.</p> <p>Assume that <code>v</code> contains <code>[1, 2, 3]</code>. The result of writing <code>cout << v[4]</code>; is a compiler error.</p> <p>The declaration: <code>vector<int> v = new vector<>()</code>; creates a vector object with no elements.</p> <p>The <code>pop_back</code> member function adds elements to the end of a vector.</p>											
Unix and C	Fortran	Simula	Berkeley Systems Distribution Unix	C++	GNU, GCC and Free Software	Ken Thomson and Dennis Ritchie	John Backus	O. Dahl & K. Nygaard	Bill Joy	Bjarne Stroustrup	Richard Stallman
Code is written in machine (and assembly) language for a specific processor; thus it is non-portable or machine dependent.				native code machine language							
Which of these statements apply to C++?				More efficient than Java or Python		Produces native code that runs on the CPU		Compiles to native code			
Converts processed source code to object code.				Compiler							
Allows you to run your program in a controlled environment.				Debugger							
Used by compiler to produce object code.				Assembler							
Combines object modules to produce an executable.				Linker							
Provides instructions for building your program.				Make							
Reads an executable image on disk and starts it running.				Loader							
Performs text substitution on your source code.				Preprocessor							
What is wrong with this IPO code fragment?				Input occurs after output							
<pre>cout << "Name: "; string name; cout << "Hello, " << name << endl; cin >> name;</pre>											



<div>cout</div> <div><<</div> <div>cin</div> <div>\n</div> <div>endl</div>	<div>Analogous to Java's System.out</div> <div>Insertion or output operator</div> <div>Similar to Java's Scanner objects</div> <div>Escape character</div> <div>Stream manipulator</div>
<div>What kind of error is this?</div> <div>error: expected ';' after expression</div>	<div>A syntax error</div>
<div>What is the problem here?</div> <div>You have submitted another student's completion code</div>	<div>You filled out the STUDENT variable incorrectly</div>
<div>What is the problem here?</div> <div>make: *** No targets specified and no makefile found. Stop.</div>	<div>The programmer is in the wrong directory.</div>
<div>The makefile for h04 is missing</div> <div>int main() { }</div>	<div>Compiles, runs and returns 0 to the O/S</div>
<div>Below is the main function from the f2c program in Chapter 1. Which line(s) contain a function declaration?</div> <div>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</div>	<div>None of these</div>
<div>Below is the main function from the f2c program in Chapter 1. Which line(s) uses the character input stream?</div> <div>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</div>	<div>Line 17</div>
<div>Below is the main function from the f2c program in Chapter 1. Which line(s) contain a function call?</div> <div>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</div>	<div>Line 18</div>
<div>Below is the main function from the f2c program in Chapter 1. Which line(s) use the insertion operator?</div> <div>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</div>	<div>Line 19</div> <div>Line 15</div>



<pre>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</pre>	
<p>Below is the main function from the f2c program in Chapter 1. Which line(s) contain a variable defintion?</p> <pre>int main() { 15 cout << "Enter a temperature in fahrenheit: "; 16 double fahr; 17 cin >> fahr; 18 double celsius = convert(fahr); 19 cout << "Converted: " << fahr << "F -> " << celsius << "C" << endl; return 0; }</pre>	<p>Line 16 line 18</p>
<p>Explain this output. Why is nothing printed?</p> <pre>#include <iostream> using namespace std; int main() { cout << "Hello, World"; } make example ./example</pre>	<p>File not saved</p>
<p>What command only builds hw04?</p>	<p>make</p>
<p>What command checks hw04 for correctness?</p>	<p>make test</p>
<p>What command hands in hw04 for course credit?</p>	<p>make submit</p>
<p>What command makes hw the current folder?</p>	<p>cd ~/workspace/cs150/hw</p>
<p>1 cout << 10 + 1 << endl;</p> <p>2 cout << (10 + 1) << endl;</p> <p>3 (cout << 11) << endl;</p>	<p>Rule 1 precedence</p> <p>Rule 2 associativity</p> <p>Rule 3 side effect</p>
<p>The _____ of an operator determines which operands the operator binds with?</p>	<p>precedence</p>
<p>The _____ of an operator determines the order of operations when operators share an operand?</p>	<p>associativity</p>
<p>The _____ of an operator determines the number of items it operates on?</p>	<p>arity</p>
<p>What prints?</p> <pre>int main() { cout << fixed << 2.0 / 3.0 << endl; }</pre>	<p>.666667</p>
<p>What prints?</p> <pre>int main() { cout << 2000 / 3.0 << endl; }</pre>	<p>6.66667e-02</p>
<p>What prints?</p> <pre>int main() { cout << fixed << 2000 / 3.0 << endl; }</pre>	<p>666.666667</p>
<p>What prints?</p> <pre>int main() { cout << setprecision(2) << 2000 / 3.0 << endl; }</pre>	<p>6.67e02</p>



<pre>int main() { cout << fixed << setprecision(2) << 2000 / 3.0 << endl; }</pre>	
<p>Match each item with the correct statement below.</p> <p>unsigned long signed int unsigned long long unsigned int signed long signed long long</p>	<p>15UL 12 15ULL 15U 3L 15LL</p>
<p>The standard input object; analogous to a Scanner in Java</p> <p>Modifies and manipulates data to produce information</p> <p>The header used to include the standard streams</p> <p>A single entity that bundles data and instructions</p> <p>Displays the results of calculations</p> <p>cout stands for _____</p> <p>The standard output object; analogous to System.out in Java</p> <p>The output or insertion operator</p> <p>endl is a _____</p> <p>Retrieves data and stores it in variables</p> <p>C++ uses an _____ library for input and output.</p> <p>Text enclosed in double quotes</p> <p>The header used for formatting real numbers</p> <p>Asking an object to perform certain operations</p>	<p>cin processing iostream object output character output cout << stream manipulator input object-oriented string iomanip sending a message</p>
<p>Literals like 3 and 7 are always:</p> <p>On line 3, b is:</p> <pre>int a = 3; // 1 int a = 7; // 2 a = b; // 3</pre>	<p>B O T H : rvalue</p>
<p>On line 2, b is:</p> <pre>int main() { a = 3; // 1 const int b = 7; // 2 a = b; // 3 }</pre>	<p>A non-modifiable lvalue</p>
<p>Which operator is the extraction operator?</p>	<p>>></p>
<p>Which operator is the insertion operator?</p>	<p><<</p>
<p>What header is needed to use the string type?</p>	<p><string></p>
<p>What header is needed to use cin and cout?</p>	<p><iostream></p>
<p>What header is needed for output formatting?</p>	<p><iomanip></p>
<p>What header is needed to use the sqrt() function?</p>	<p><cmath></p>
<p>Which line prints 5?</p> <pre>int n = 12; cout << n/3 << endl; // 1 cout << n/7 << endl; // 2 cout << n % 3 << endl; // 3 cout << n % 7 << endl; // 4</pre>	<p>4</p>

Total combine1		Study	<div></div>
<pre>int sum = 22; sum +=2; cout << sum++; // sum = sum + 4 cout << sum << endl;</pre>			
<p>What is the output of the following program?</p> <pre>int value = 3; value++; cout << value << endl;</pre>		4	
<p>Which line or lines are illegal?</p> <pre>/1/ int a, b; /2/ a = 3; int main() { /3/ b = 4; /4/ cout << a << ", " << b << endl ; }</pre>		2	
<p>Which line or lines are illegal?</p> <pre>int a; int b = 3; int main() { a = 4; cout << a << ", " << b << endl; }</pre>		None of these	
<p>Symbols which directly represent a value</p> <p>Determines the direction of operations for operators at the same level</p> <p>Describes how many operands an operator requires</p> <p>Operators that require a single data value</p> <p>Symbol which indicates a value</p> <p>Determines how tightly operators bind to operands</p> <p>Any combination of operators and operands which yields a value</p> <p>A symbol that can be used to produce a value at runtime</p> <p>Symbol which indicates an operation</p> <p>A storage location containing a value</p> <p>Operators that require two data values</p>		literal	associativity
		arity	unary
		operand	precedence
		expression	function call
		operator	variable
		binary	
<p>Types such as classes, structures and enumerations</p> <p>Types such as pointers, arrays and references</p> <p>Built-in types, such as int and double</p> <p>The "kind" of a variable</p> <p>Read a value and store it in a variable</p> <p>Types such as string and vector</p>		user-defined types	derived types
		primitive types	data type
		input	library types
<p>Which of these five concepts are illustrated here?</p> <pre>int main() { int a; }</pre>		Declaration	Definition
<p>Which of these five concepts are illustrated here?</p> <pre>int main() { extern int a; }</pre>		Declaration	Definition

Total combine1		Study	<div>1/1</div>
<pre>int main() { extern int a; a = 3; }</pre>			
Associates a name with a type	declare		
Read a value and store it in a variable	input		
Copy a new value into an existing variable	assign		
Allocates space for a variable	define		
Provides a starting value when a variable is created	initialize		
A named storage area that holds a value	variable		
What is true about identifiers in C++?	They may contain an underscore		
As an application programmer, which of the following names for local variables are both legal and recommended for stylistic reasons.	<div>_(single underscore)</div> <div>CamelCase</div>		
As an application programmer, which of the following names for local variables are legal (even if they are unwise from a stylistic perspective).	<div>cout</div> <div>2cool</div> <div>CamelCase</div> <div>u2</div> <div>integer</div>		
Which manipulator is used to ensure that large numbers appear using regular decimal notation?	fixed		
Which manipulator is used to change the padding character used in a column like: 0045?	setfill()		
Which manipulator(s) is/are used to make sure the value 2.0/3 prints like this: 0.677?	<div>fixed</div> <div>setprecision()</div>		
Which manipulator(s) is/are used to make sure the number 45 prints like this: 0045?	<div>setfill()</div> <div>setw()</div>		
<div>Assume int x, y, z;</div> <div>Shorthand assignment</div> <div>Post increment</div> <div>Undefined behavior</div> <div>Widening conversion</div> <div>Pre decrement</div> <div>Chained assignment</div> <div>Narrowing conversion</div> <div>Mixed-type expression</div>	<div>y += z;</div> <div>x++;</div> <div>x = z++ - ++z;</div> <div>double a = y;</div> <div>--z;</div> <div>x = y = z = 10;</div> <div>z = 3.15;</div> <div>auto v = x * 2.3;</div>		
<div>Which of the following variables have the value 0?</div> <div>int global;</div> <div>int main</div> <div>{</div> <div>string localStr;</div> <div>double localDouble;</div> <div>}</div>	global		
<div>Which of the following variables have an undefined value?</div> <div>int global;</div> <div>int main</div> <div>{</div> <div>string localStr;</div> <div>double localDouble;</div> <div>}</div>	localDouble		
<div>Which of the following variables have the value null?</div> <div>int global;</div> <div>int main</div> <div>{</div> <div>string localStr;</div> <div>double localDouble;</div> <div>}</div>	None of these		
The variable ASSIGNMENT from your homework has been _____.	declared		



	declared
<p>This code is legal, compiles and is well defined. Which line(s) contain an assignment?</p> <pre>int a = 5; // 1 a == 5; // 2 int b = 6; // 3 a ={b}; // 4 auto c = a == b; // 5</pre>	4
<p>This code is legal, compiles and is well defined. Which line(s) contain comparison?</p> <pre>int a = 5; // 1 a == 5; // 2 int b = 6; // 3 a ={b}; // 4 auto c = a == b; // 5</pre>	2 5
<p>This code is legal, compiles and is well defined. Which line(s) contain initialization?</p> <pre>int a = 5; // 1 a == 5; // 2 int b = 6; // 3 a ={b}; // 4 auto c = a == b; // 5</pre>	1 3 5
<p>This code is legal, compiles and is well defined. Which line(s) contain an input statement?</p> <pre>int a = 5; // 1 a == 5; // 2 int b = 6; // 3 a ={b}; // 4 auto c = a == b; // 5</pre>	None of these
The + arithmetic operator is a(n) _____ operator	binary
The - operator is a(n) _____ operator	unary binary
The ++ arithmetic operator is a(n) _____ operator	side effect unary
A set of bits interpreted according to its type	Value
x in the expression x = 3;	lvalue
x in the expression y = x;	rvalue
Uniform or list initialization	int c{5};
Legacy or assignment initialization	int a = 0;
Direct initialization	int b(3);
const double PI = 3.14159;	non-modifiable value
narrowing conversion	int e(3.5);
<p>Assume that the user enters: Steve 60 3.5 What value is stored in gpa?</p> <pre>string name; int age; double gpa; cout << "Enter your name, age and gpa: "; cin >> name >> age >> gpa;</pre>	3.5
<p>Assume that the user enters: Steve 3.5 68 What value is stored in gpa?</p> <pre>string name; int age; double gpa; cout << "Enter your name, age and gpa: "; cin >> name >> age >> gpa;</pre>	.5



<pre>string name; int age; double gpa; cout << "Enter your name, age and gpa: "; cin >> name >> age >> gpa;</pre>	
<p>Assume that the user enters: Steve Gilbert 68 3.5 What value is stored in age?</p> <pre>string name; int age; double gpa; cout << "Enter your name, age and gpa: "; cin >> name >> age >> gpa;</pre>	undefined
<p>Which of these are impossible conditions?</p> <pre>auto floor ??? // some number; bool v1 = floor >= 0 floor <= 20; bool v2 = floor <= 0 && floor >= 20; bool v3 = floor <= 0 floor >= 20; bool v4 = floor >= 0 && floor <= 20; bool v5 = floor >= 0 floor < 20; bool v6 = floor >= 0 && floor > 20; bool v7 = floor >= 0 floor > 20; bool v8 = floor >= 0 && floor < 20;</pre>	v2
<p>Which of these are unavoidable conditions?</p> <pre>auto floor ??? // some number; bool v1 = floor >= 0 floor <= 20; bool v2 = floor <= 0 && floor >= 20; bool v3 = floor <= 0 floor >= 20; bool v4 = floor >= 0 && floor <= 20; bool v5 = floor >= 0 floor < 20; bool v6 = floor >= 0 && floor > 20; bool v7 = floor >= 0 floor > 20; bool v8 = floor >= 0 && floor < 20;</pre>	v1 v5
<p>[] and () denote whether a range includes or excludes an endpoint: [includes the endpoint (excludes the endpoint [] = 'Closed', includes both endpoints () = 'Open', excludes both endpoints []) and ([) are both 'half-open', and include only one endpoint</p> <p>Which variable correctly indicates that the variable floor is in the range [0...20)?</p> <pre>auto floor ??? // some number; bool v1 = floor >= 0 floor <= 20; bool v2 = floor <= 0 && floor >= 20; bool v3 = floor <= 0 floor >= 20; bool v4 = floor >= 0 && floor <= 20; bool v5 = floor >= 0 floor < 20; bool v6 = floor >= 0 && floor > 20; bool v7 = floor >= 0 floor > 20; bool v8 = floor >= 0 && floor < 20;</pre>	v8
<p>[] and () denote whether a range includes or excludes an endpoint: [includes the endpoint (excludes the endpoint [] = 'Closed', includes both endpoints () = 'Open', excludes both endpoints []) and ([) are both 'half-open', and include only one endpoint</p> <p>Which variable correctly indicates that the variable floor is in the range (0...20)?</p> <pre>auto floor ??? // some number; bool v1 = floor >= 0 floor <= 20; bool v2 = floor <= 0 && floor >= 20; bool v3 = floor <= 0 floor >= 20; bool v4 = floor >= 0 && floor <= 20; bool v5 = floor >= 0 floor < 20; bool v6 = floor >= 0 && floor > 20; bool v7 = floor >= 0 floor > 20; bool v8 = floor >= 0 && floor < 20;</pre>	v3

Total combine1		Study	<div></div>
<div>(excludes the endpoint [] = 'Closed', includes both endpoints () = 'Open', excludes both endpoints [] and () are both 'half-open', and include only one endpoint</div> <div>Which variable correctly indicates that the variable floor is in the range [0...20]?</div> <div>auto floor ??? // some number; bool v1 = floor >= 0 floor <= 20; bool v2 = floor <= 0 && floor >= 20; bool v3 = floor <= 0 floor >= 20; bool v4 = floor >= 0 && floor <= 20; bool v5 = floor >= 0 floor < 20; bool v6 = floor >= 0 && floor > 20; bool v7 = floor >= 0 floor > 20; bool v8 = floor >= 0 && floor < 20;</div>			
<div>Strings in C++ are mutable.</div> <div>String in C++ are immutable</div>		<div>True</div> <div>False</div>	
<div>In C++ you can compare strings using all of the relational operators.</div> <div>In C++ you cannot use the relational or equality operators with strings.</div>		<div>True</div> <div>False</div>	
<div>Assuming str is a string object, this syntax is legal in both Java and C++. Does this code work correctly in both languages?</div> <div>if (str == "quit") . . .</div>		<div>False</div>	
<div>In C++ you can concatenate string objects using the + or += operators.</div>		<div>True</div>	
<div>Assuming str is a string object, is this syntax legal in both Java and C++?</div> <div>if (str == "quit") . . .</div>		<div>True</div>	
<div>Assuming str is a string object does this correctly test if str consists of the characters "quit" in C++?</div> <div>if (str == "quit") . . .</div>		<div>True</div>	
<div>Assuming lastName is a string object, does this work as expected in C++?</div> <div>if (lastName <= "Gilbert") . . .</div>		<div>True</div>	
fruitful function		A function that calculates and returns a value	
body		A block containing statements that implement the function's actions.	
function		A named block of code that carries out an action or calculates a value.	
prototype		Another name for a function declaration	
parameters		Variables defined along with the function to receive input	
calling		Executing, running or invoking the function	
procedure		A function that carries out an action instead of calculating a value.	
return statement		Produces a value when the function is invoked	
defining		Specifying the calculation or actions that occur when the function is used	
declaring		Specifying the function name, type and parameter types.	
arguments		Values passed to the function when it is invoked	
<div>Which control structure is best equipped to handle processing for a group of check boxes?</div>		<div>independent if statements</div>	
<div>Which control structure is best equipped to handle an on or off condition?</div>		<div>if-else statements</div>	
<div>Which control structure is best equipped to handle numeric selections made from a menu?</div>		<div>the switch statement</div>	
<div>Which control structure is best equipped to handle processing for a group of radio buttons?</div>		<div>sequential if statements</div>	
<div>Which control structure is best equipped to handle processing for income taxes?</div>		<div>nested if statements</div>	
<div>Which control structure is best equipped to set a variable to one or two possible values?</div>		<div>the conditional operator</div>	

Total combine1		Study	<div></div>
<pre>if (n % 2 == 1) cout << "Odd" << endl; else cout << "Even" << endl;</pre>			
<p>This code illustrates the ____ idiom.</p> <pre>if (n % 2 == 1) cout << "Odd" << endl;</pre>	guarded action		
<p>This code illustrates the ____ idiom.</p> <pre>auto n = 3; if (n % 2 == 1) n = -n; else if (n < 0) n++; else if (n % 2 = 0) n--; else n = 0;</pre>	multiple selection		
<p>This code illustrates the ____ idiom.</p> <pre>auto n = 3; if (n % 2 == 1) n = -n; if (n < 0) n++; if (n % 2 = 0) n--;</pre>	Independent if statements		
<p>This code illustrates the ____ idiom.</p> <pre>auto n = 3; else if (n % 2 == 1) n = -n; else if (n < 0) n++; else if (n % 2 = 0) n--; else n = 0;</pre>	None of these are correct		
The C++ string class is defined in the header:	<string>		
You can find the length of a string str using str.size(). In C++, size() is called:	a member function		
Which operator is used to see if all of a set of conditions is true?	logical and		
Which operator is used to see if any of a set of conditions is true?	logical or		
<p>If a is false, which expressions need not be evaluated?</p> <pre>if (a && b c && d e) ...</pre>	b		
<p>If a and c are both false, which expressions need not be evaluated?</p> <pre>if (a && b c && d e) ...</pre>	b, d		
<p>If a and b are true, which expressions need not be evaluated?</p> <pre>if (a && b c && d e) ...</pre>	c, d, e		
Produces the empty string	string s1; (choice A)		
Implicitly converts a character array to a string object	string s2 = "hello"; (choice B)		
Explicitly converts a character array to a string object	string s3{"world"}; (choice C)		
Produces a string from multiple copies of a single character	string s4(20, '-'); (choice D)		
Produces a string that may contain quotes or backslashes	string s5(R("bob")); (choice E)		
Needed to use the C++ string type	#include <string> (choice F)		
Reads one word or token from standard input	cin >> s1; (choice G)		
Reads one line of text from standard input	getline(cin, s2); (choice H)		
<p>Assume the user enters Inigo Montoya when prompted. What prints?</p> <pre>cout << "What's your name: "; string name; cin >> name; cout << "Howdy " + name + "!";</pre>	Howdy Inigo!		
<p>What is the output?</p> <pre>auto x = 40; if (x <= 40) cout << "F"; if (x <= 75) cout << "C"; if (x <= 90) cout << "B"; cout << endl;</pre>	FCB		
In C++, what is true about concatenating string literals (character arrays)?	you do it by separating the string literals with white space		
In C++, the statement string s{3, 'X'};	creates a string variable of size 3, filled with 'X'		

Total combine1		Study	
In C++, the statement string s = "world";		creates a string variable implicitly initialized with the character array "world"	
In C++, what keyword is used for type inference?		auto	
In C++, characters of the type char:		Can be preceded by signed or unsigned for use as small integers Generally use 8 bits of storage. Use the ASCII character set Are defined for the first 127 characters	
In C++, characters of type char:		Can be preceded by signed or unsigned for use as small integers Generally use 8 bits of storage. Use the ASCII character set Are defined for the first 127 characters	
In C++, what is true about the += operator operating on string objects		You may concatenate creates a string variable to a string object You may concatenate a string literal (character array) to a string object You may concatenate a char literal to a string object You may concatenate a char variable to a string object	
The code shown here: auto n = 3; if (n = 0) cout << "n is 0" << endl; else cout << "n is " << n << endl;		Executes the false branch Displays "n is 0" Contains an embedded assignment	
Assume that name is a string object. Which of these expressions are legal?		name += 'X' name < "bob" name == "sally" name += "fred"	
What is true about string::size_type?		It is the same as size_t It is returned from the string size() member function It is returned from the string length() member function It is an unsigned integer type of some size You may create variables of that type	
Compare C++ and Java string. Which of these are true?		"hello" is a string object in Java, but not in C++ String s; produces the null string in Java, while string s; produces the empty string in C++. String is capitalized in Java, lowercase in C++ Assuming str is a string, str * "b" is legal in both Java and C++	
What header file do you include to call the isupper() function?		<cctype>	
All of these are declared in the <string> header; which are member functions?		size() front() find() at()	
Match the letter of the variable in the figure with the correct value or expression below string s{"walk the plank"}; auto a = s.find('a'); auto b = s.find('a', 3); auto c = s.find("nk"); auto d = s.find("Walk");		a : 1 b : 11 c : 12 d : string::npos	
One-way, independent decisions use:		if	
Either/or decisions should use:		if ... else	
Multiple possible outputs, testing a single condition, use:		if ... else ... if ... else	
Leveled decisions, such as processing income taxes are best handled with:		if ... if ... else ... else	
To produce one of two values (of any type) in an expression, use:		a conditional operator	
To combine several test conditions to produce a single Boolean value, use:		a logical operator	
In Line 2, what is the receiver? string s{"happy"}; auto pos = s.find('y');		s	
Decisions based on numbered blocks of code are best handled with:		switch	
In Line 2, what is the result of this function call? string s{"happy"}; auto pos = s.find('y');		pos	



<pre>string s{"happy"}; auto pos = s.find('y');</pre>	
<p>In Line 2, what is the explicit argument?</p> <pre>string s{"happy"}; auto pos = s.find('y');</pre>	'y'
<p>In Line 2, what is the implicit argument?</p> <pre>string s{"happy"}; auto pos = s.find('y');</pre>	the address of s
<p>In Line 2, what is the parameter?</p> <pre>string s{"happy"}; auto pos = s.find('y');</pre>	None of these
<p>Assume c is a char variable. What type is the variable a?</p> <pre>string s{"guten tag"}; auto len = s.size(); auto a = s.front(); s.at(len) = a; s[len] = c;</pre>	char
<p>Assume c is a char variable. What value s stored in the variable a?</p> <pre>string s{"guten tag"}; auto len = s.size(); auto a = s.front(); s.at(len) = a; s[len] = c;</pre>	'g'
<p>What type is the variable len?</p> <pre>string s{"guten tag"}; auto len = s.size(); auto a = s.front(); s.at(len) = a; s[len] = c;</pre>	string::size_type
<p>Assume c is a char variable. What type is the expression s.last()?</p> <pre>string s{"guten tag"}; auto len = s.size(); auto a = s.front(); s.at(len) = a; s[len] = c;</pre>	None of these
<p>The relative order of two variables is tested using:</p>	a relational operator
<p>Assume c is a char variable. Which line throws an error because of range checking?</p> <pre>string s{"guten tag"}; // 1 auto len = s.size(); // 2 auto a = s.front(); // 3 s.at(len) = a; // 4 s[len] = c; // 5</pre>	4
<p>Assume c is a char variable. Which line produces undefined behavior?</p> <pre>string s{"guten tag"}; // 1 auto len = s.size(); // 2 auto a = s.front(); // 3 s.at(len) = a; // 4 s[len] = c; // 5</pre>	5
<p>Assume c is a char variable. Which line produces a syntax error?</p> <pre>string s{"guten tag"}; // 1 auto len = s.size(); // 2 auto a = s.front(); // 3 s.at(len) = a; // 4 s[len] = c; // 5</pre>	None of these
<p>The string find() member function may be used to search for a substring</p> <p>The string find() member function takes either a string or character as an argument</p> <p>The string find() member function throws an exception if the target cannot be found.</p>	<p>True</p> <p>True</p> <p>False</p>



Calling s.at(1) returns a copy of the second character in the string object s	False
s.at(0) = 'c'; changes the first character in the string object s to 'c'	True
s.at(0) = "c"; changes the first character in the string object s to 'c'	False
The getline() function is part of the string class.	False
Data member is the term used in C++ for what is called a method in Java	False
The toupper() member function ignores case when it searches.	False
In C++ a char may be one, two or three bytes, when using UTF-8.	False
s.back() = 'x'; changes the last character in the string object s to 'x'.	True
Calling s.at(0) returns the same reference as s.front().	True
A C++ string that contains Unicode characters should be preceded by:	u8
To enter a Unicode character into a C++ string, use an escape sequence starting with:	\U
Which of these selects a character (char) from a string?	auto a = s[0];
This compiles, runs and prints 12. What is the correct parameter declaration for x?	int& x
This compiles, runs and prints 4, 3. What is the correct prototype? int x = 3, y = 4; swap(x, y); cout << x << ", " << y << endl;	void swap(int& a, int& b);
What value is stored in a after this runs? string s{"ABCDEFD"}; auto a = s.find('D');	3
What value is stored in a after this runs? string s{"abcdefg"}; auto a = s.substr(3);	"defg"
What value is stored in a after this runs? string s{"ABC"}; auto a = s.substr(4, 5);	Runtime error because start (4) must bet 0..3
What value is stored in a after this runs? string s{"ABCDEFD"}; auto a = s.find('G');	string::npos
What value is stored in a after this runs? string s{"ABCDEFGHIJKLM"}; auto a = s.substr(1, 4);	"BCDE"
String parameters should be passed to functions:	by constant reference (const string& s) when not modified in the function. by reference (string& s) when modified in the function
Assume a is 5 and b is 3; what prints? string s{"feed the fish"}; cout << s.substr(a, b) << endl;	"the"
Assume a is 9 and b is 10; what prints? string s{"feed the fish"}; cout << s.substr(a, b) << endl;	"fish"
Assume a is 13 and b is 10; what prints? string s{"feed the fish"}; cout << s.substr(a, b) << endl;	""
Assume a is 14 and b is 10; what prints? string s{"feed the fish"}; cout << s.substr(a, b) << endl;	Runtime error



<pre>int n = 4; int& r1 = n; auto& r2 = r1; r1 = 3; r2 = 5; cout << n << endl;</pre>	
<p>What does this code segment print?</p> <pre>int n = 4; int& r1 = n; auto& r2 = r1; r1 = 3; r2 = 5; cout << n << endl;</pre>	5
<p>Which of these lines are illegal?</p> <pre>[1] int n1 = 4; [2] double n2 = 3.145; [3] unsigned char n3 = 158; [4] int n4 = n2; [5] int& r1 = n1; [6] int& r2 = n2; [7] double& r3 = n1; [8] const int& r4 = n2;</pre>	6 7
<p>Which of these lines are legal?</p> <pre>[1] int n1 = 4; [2] double n2 = 3.145; [3] unsigned char n3 = 158; [4] int n4 = n2; [5] int& r1 = n1; [6] int& r2 = n2; [7] double& r3 = n1; [8] const int& r4 = n2;</pre>	4 5 8
<p>Which lines cause runtime errors (exceptions)?</p> <pre>[1] string s{"shiver me timbers"}; [2] auto len = s.size(); [3] s.front() = 'S'; [4] s.back() = "S"; [5] s[len] = 'X'; [6] s.substr(0, 1) = "W"; [7] auto a = s.substr(0, 100); [8] auto b = s.substr(4, 3); [9] auto c = s.substr(len);</pre>	None of these
<p>Which lines compile and return string objects?</p> <pre>[1] string s{"shiver me timbers"}; [2] auto len = s.size(); [3] s.front() = 'S'; [4] s.back() = "S"; [5] s[len] = 'X'; [6] s.substr(0, 1) = "W"; [7] auto a = s.substr(0, 100); [8] auto b = s.substr(4, 3); [9] auto c = s.substr(len);</pre>	7 8 9
<p>Which lines cause syntax errors?</p> <pre>[1] string s{"shiver me timbers"}; [2] auto len = s.size(); [3] s.front() = 'S'; [4] s.back() = "S"; [5] s[len] = 'X'; [6] s.substr(0, 1) = "W"; [7] auto a = s.substr(0, 100); [8] auto b = s.substr(4, 3); [9] auto c = s.substr(len);</pre>	4 6
<p>What is stored in s after this code runs?</p> <pre>string s{"xyzw"}; s.at(2) = 'Y';</pre>	xyYw
<pre>string s{"ahoy"}; auto a = s.size(); auto b = s.back(); auto c = s.at(0); auto d = s.substr(a); auto e = s.substr(0, 1);</pre>	[a] : string::size_type [b] : 'y' [c] : 'a' [d] : "" [e] : "a"

Total combine1		Study	
[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?		[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 occurring 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	
Look at the problem statement below. The _____ of the loop is to count the number of characters in a sentence. [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.]		goal	
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	



<pre>for (int i = 0; i < 10; i++) cout << i; cout << endl;</pre>	
<p>How many times is this loop entered? (That is, how many times is i printed?)</p> <pre>for (int i = 0; i <= 10; i++) cout << i; cout << endl;</pre>	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
<p>Below is the illustration from the loop building strategy in Chapter 5. The highlighted lines represents:</p> <p>Given: the variable str is a string (may be empty) Create the counter variable, initialized to -1</p> <pre> 🌟 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</pre>	a loop guard
<p>Below is the illustration from the loop building strategy in Chapter 5. The highlighted lines represents:</p> <p>Given: the variable str is a string (may be empty) Create the counter variable, initialized to -1</p> <p>If the variable str has any characters then</p> <pre>{ 🌟 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</pre>	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



<p>Given: the variable str is a string (may be empty) Create the counter variable, initialized to -1</p> <p>If the variable str has any characters then {</p> <p>Set counter to 0 Create the variable current-character as a character Place the first character in str into current-character</p> <p>While more-characters and current-character not a period { Add one to (or increment) the counter variable</p> <p>☀ Store the next character from str in current-character ☀</p> <p>}</p> <p>If current-character is a period then Add one to the counter to account for the period. Else Set counter to -2</p> <p>}</p> <p>If counter is -1 the string was empty Else if counter is -2 there was no period</p>	
<p>Below is the illustration from the loop building strategy in Chapter 5. The highlighted lines represents:</p> <p>Given: the variable str is a string (may be empty) Create the counter variable, initialized to -1</p> <p>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</p> <p>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 }</p> <p>☀ If current-character is a period then ☀</p> <p>Add one to the counter to account for the period. Else Set counter to -2]</p> <p>}</p> <p>If counter is -1 the string was empty Else if counter is -2 there was no period</p>	<p>loop postcondition</p>
<p>In a guarded loop, the loop actions may never be executed</p> <p>In a guarded loop, the loop actions are always executed at least once.</p>	<p>True</p> <p>False</p>
<p>In an unguarded loop, the loop actions are always executed at least once.</p> <p>In an unguarded loop, the loop actions may never be executed.</p>	<p>True</p> <p>False</p>
<p>A guarded loop is also known as a test-at-the-top loop</p> <p>A guarded loop is also known as a test-at-the-bottom loop.</p>	<p>True</p> <p>False</p>
<p>An unguarded loop is also known as a test-at-the-bottom loop.</p> <p>An unguarded loop is also known as a test-at-the-top loop.</p>	<p>True</p> <p>False</p>
<p>Loops are used to implement iteration in C++.</p> <p>Loops are used to implement selection in C++.</p>	<p>True</p> <p>False</p>



<pre>for (int i = 1; i <= 10; i++) cout << i; cout << endl;</pre> <p>This idiomatic pattern is used to count from one value to another.</p> <pre>for (int i = 1; i < 10; i++) cout << i; cout << endl;</pre>	False
<p>This loop uses asymmetric bounds.</p> <pre>for (int i = 0; i < 10; i++) cout << i; cout << endl;</pre> <p>This loop uses asymmetric bounds.</p> <pre>for (int i = 1; i < 10; i++) cout << i; cout << endl;</pre> <p>This loop uses asymmetric bounds.</p> <pre>for (int i = 1; i <= 10; i++) cout << i; cout << endl;</pre>	<p>True</p> <p>True</p> <p>False</p>
<p>[1301] Which line below points ppi to pi?</p> <pre>int main() { double pi = 3.14159; double *ppi; // code goes here // code goes here }</pre>	<pre>ppi = &pi;</pre>
<p>[1302] Assume that ppi correctly points to pi. Which line prints the value stored inside pi?</p> <pre>int main() { double pi = 3.14159; double *ppi; // code goes here // code goes here }</pre>	<pre>cout << &pi; cout << ppi; cout << &ppi; cout << *pi;</pre> <p>→ None of these</p>
<p>[1303] Assume that ppi correctly points to pi. Which line prints the value stored inside pi?</p> <pre>int main() { double pi = 3.14159; double *ppi; // code goes here // code goes here }</pre>	<pre>cout << *ppi;</pre>
<p>[1304] Assume that ppi correctly points to pi. Which line prints the address of ppi?</p> <pre>int main() { double pi = 3.14159; double *ppi; // code goes here // code goes here }</pre>	<pre>cout << &ppi;</pre>
<p>[1305] Assume that ppi correctly points to pi. Which line prints the size (in bytes) of pi?</p> <pre>int main() { double pi = 3.14159; double *ppi; // code goes here // code goes here }</pre>	<pre>cout << sizeof(*ppi);</pre>

Total combine1		Study	...
<pre>int a = 1; void f(int b) { int c = 3; static int d = 4; }</pre>			
<p>[1307] The value for the variable b is stored:</p> <pre>int a = 1; void f(int b) { int c = 3; static int d = 4; }</pre>		on the stack	
<p>[1308] The value for the variable c is stored:</p> <pre>int a = 1; void f(int b) { int c = 3; static int d = 4; }</pre>		on the stack	
<p>[1309] The value for the variable d is stored:</p> <pre>int a = 1; void f(int b) { int c = 3; static int d = 4; }</pre>		in the static storage area	
<p>[1310] The variable buf is a pointer to a region of memory storing contiguous int values. (This is similar to your homework, where you had a region of memory storing unsigned char values) The four lines shown here are legal. Which operation is illegal?</p> <pre>int *p1 = buf; const int *p2 = buf; int * const p3 = buf; const int * p4 const = buf;</pre> <p>p2++; *p1 = 3; *p3 = 5; p1++; *p2 = 7</p>		*p2 = 7;	
<p>[1311] The variable buf is a pointer to a region of memory storing contiguous int values. (This is similar to your homework, where you had a region of memory storing unsigned char values.) The four lines shown here are legal. Which operation is illegal?</p> <pre>int *p1 = buf; const int *p2 = buf; int * const p3 = buf; const int * p4 const = buf;</pre>		p3++;	
<p>[1312] The variable buf is a pointer to a region of memory storing contiguous int values. (This is similar to your homework, where you had a region of memory storing unsigned char values.) The four lines shown here are legal. Which operation is legal?</p> <pre>int *p1 = buf; const int *p2 = buf; int * const p3 = buf; const int * p4 const = buf;</pre>		*p3 = 5;	
<p>[1313] These pointer should point to "nothing". Which is not correctly initialized?</p>		vector<int> *vp;	
<p>[1314] These pointer should point to "nothing". Which is not correctly initialized?</p> <pre>Star *ps = NULL; vector<int> *vp(0); int *pi = nullptr; double *pd{}; All are correctly initialized to point to nothing</pre>		All are correctly initialized to point to nothing	
<p>[1315] Which of these is the preferred way to initialize a pointer so that it points to "nothing"?</p>		int *pi = nullptr;	



<pre>int a = 3, b = 4;</pre>	
<p>[1318] All of these are legal C++ statements; which of them uses the C++ reference declarator?</p> <pre>int a = 3, b = 4;</pre>	<pre>int &x = a;</pre>
<p>[1319] All of these are legal C++ statements; which of them uses the C++ pointer declarator?</p> <pre>int a = 3, b = 4;</pre>	<pre>int *p = &b;</pre>
<p>[1320] All of these are legal C++ statements; which of them uses the C++ dereferencing operator?</p> <pre>int a = 3, b = 4;</pre>	<pre>int x = *p;</pre>
<p>[1321] All of these are legal C++ statements; which of them uses indirection?</p> <pre>int a = 3, b = 4;</pre>	<pre>int x = *p;</pre>
<p>[1322] In C++, global variables are stored:</p>	in the static storage area
<p>[1323] What is true about an uninitialized pointer?</p>	Dereferencing it is undefined behavior
<p>[1324] What is true about this code?</p> <pre>int n{500}; int *p = &n;</pre>	*p is the value of n
<p>[1325] What is true about this code?</p> <pre>int * choice;</pre>	choice contains an undefined address
<p>[1326] How can we print the address where n is located in memory?</p> <pre>int n{500};</pre>	<pre>cout << &n << endl;</pre>
<p>[1327] Which expression obtains the value that p points to?</p> <pre>int x{100}; int *p = &x;</pre>	*p
<p>[1328] What is a common pointer error?</p>	Using a pointer without first initializing it
<p>[1329] What is printed when you run this code?</p> <pre>int x{100}; cout << &x << endl;</pre>	The memory location where x is stored
<p>[1330] What is printed when you run this code?</p> <pre>int n{}; int *p = &n; *p = 10; n = 20; cout << *p << endl;</pre>	20
<p>[1331] What is printed when you run this code?</p> <pre>int num = 0; int *ptr = &num; num = 5; *ptr += 5; cout << num << " " << *ptr << endl;</pre>	10 10
<p>[1332] What is printed when you run this code?</p> <pre>int *n{nullptr}; cout << n << endl;</pre>	The address value 0
<p>[1333] What is printed when you run this code?</p> <pre>int *n{nullptr}; cout << *n << endl;</pre>	No compilation errors, but undefined behavior
<p>[1334] What is printed when you run this code?</p> <pre>int *n{nullptr}; cout << &n << endl;</pre>	The address value where n is stored
<p>[1335] What is printed when you run this code?</p> <pre>int *p = &0; cout << *p << endl;</pre>	No output; compiler error.

Total combine1		Study	<div></div>
<pre>int n{}; int *p; *p = &n; cout << *p << endl;</pre>			
<p>[1337] What is printed when you run this code?</p> <pre>int n{}; int *p; *p = n; cout << *p << endl;</pre>	No compilation errors, but undefined behavior when run		
[1338] What is the term used to describe a variable with stores a memory address?	pointer		
[1339] Which of these is not one of the three characteristics of every variable?	alias		
[1340] Which area of memory is your program code stored in?	Text		
[1341] Which area of memory are local variables stored in?	Stack		
[1342] Which area of memory are global variables stored in?	Static storage area		
<p>[1343] Examine the following code. What is stored in c after it runs.</p> <pre>int f(int * p, int x) { *p = x * 2; return x / 2; } ... int a = 3, b, c; c = f(&b, a);</pre>	1		
<p>[1344] Examine the following code. What is stored in b after it runs.</p> <pre>int f(int * p, int x) { *p = x * 2; return x / 2; } ... int a = 3, b, c; c = f(&b, a);</pre>	6		
<p>[1345] Examine the following code. What is stored in a after it runs.</p> <pre>int f(int * p, int x) { *p = x * 2; return x / 2; } ... int a = 3, b, c; c = f(&b, a);</pre>	3		
<p>[1346] Examine this version of the swap() function, which is different than the two versions appearing in your text. How do you call it?</p> <pre>void swap(int& x, int * y) { ... } ... int a = 3, b = 7; // What goes here ?</pre>	swap(a, &b);		
<p>[1347] Examine this version of the swap() function, which is different than the two versions appearing in your text. How do you call it?</p> <pre>void swap(int * x, int & y) { ... } ... int a = 3, b = 7; // What goes here ?</pre>	swap(&a, b);		
[1348] Assume that p is a pointer to the first of 50 contiguous integers stored in memory. What is the address of the first integer appearing after this sequence of integers?	p + 50;		
[1349] Assume that p1 is a pointer to an integer and p2 is a pointer to a second integer. Both integers appear inside a large contiguous sequence in memory, with p2 storing a larger address. How many total integers are there in the slice between p1 and p2?	p2 - p1;		

Total combine1		Study	<div></div>
Let p point the beginning of the image Set end to point just past the end While p != end If *(p + 3) is 0 (transparent) Clear all of the fields Increment p by 4			
[1351] Here is a fragment of pseudocode for the negative() function in H12. What statement represents the underlined portion of code? Let p point to beginning of the image Let end be pixel one past the end of the image While p != end Invert the red component Move p to next component		p++;	
Used to access the data inside a variable Determines the amount of memory required and the operations permitted on a variable The meaning assigned to a set of bits stored at a memory location An object whose value is an address in memory Expression using the address operator Expression using the reference declarator Expression using the dereferencing operator Expression using the pointer declarator Expression returning the number of allocated bytes used by an object Address value 0		variable name variable type variable value pointer p = &a; int x = 3; y = *a; double * v; sizeof(Star) nullptr	
[1401] Which of these lines correctly prints 3? struct S { int a = 3; double b = 2.5; }; S obj, *p = &obj; cout << p.a << endl; cout << *p.a << endl; cout << *(p).a << endl; cout << *(p.a) << endl; cout << (*p).a << endl;		cout << (*p).a << endl;	
[1402] Which of these lines correctly prints 2.5? struct S { int a = 3; double b = 2.5; }; S obj, *p = &obj; cout << *(p).b << endl; cout << *p.b << endl; cout << p->b << endl; cout << *(p.b) << endl; cout << *p->b << endl;		cout << p->b << endl;	
[1403] Which of these lines displays the eighth element of a? int a[15]; cout << a[8] << endl; cout << a(7) << endl; cout << a.at(7) << endl; cout << a[7] << endl;		cout << a[7] << endl;	
[1404] Which prints the number of elements in a? int a[] = {1, 2, 3}; cout << a.length << endl; cout << sizeof(a[0]) << endl; cout << a.size() << endl; cout << sizeof(a) << endl; None of these		None of these	



<pre>int nums[3] = {1, 2}; Undefined value 2 Syntax error in array declaration 0 1</pre>	
<p>[1406] Which line throws an out_of_range exception?</p> <pre>double speed[5] = { . . . }; None of these cout << speed[4] << endl; cout << speed[5] << endl; cout << speed[0] << endl; cout << speed[1] << endl;</pre>	None of these
<p>[1407] Which line has undefined output?</p> <pre>double speed[5] = { . . . }; cout << speed[5] << endl; cout << speed[0] << endl; None of these cout << speed[1] << endl; cout << speed[4] << endl;</pre>	cout << speed[5] << endl;
<p>[1408] Which line creates an array with 5 elements?</p> <pre>int[5] d; int b[5]; int a[4]; None of these int[] c[5];</pre>	int b[5];
<p>[1409] What is printed?</p> <pre>int a[] = {1, 2, 3}; int b[] = {1, 2, 3}; if (a == b) cout << "a == b" << endl; else cout << "a != b" << endl; a != b Undefined behavior a == b Syntax error; does not compile.</pre>	a != b
<p>[1410] What does the array a contain after this runs?</p> <pre>int a[] = {1, 2, 3}; int b[] = {4, 5, 6}; a = b; Syntax error; does not compile. {4, 5, 6} {1, 2, 3} Undefined behavior</pre>	Syntax error; does not compile.
<p>[1411] Which assigns a value to the first position in letters?</p> <pre>char letters[26]; letters[0] = 'a'; letters[0] = "a"; letters[1] = 'b'; letters.front() = 'a'; letters = 'a';</pre>	letters[0] = 'a';
<p>[1412] Which assigns a value to the first position in letters?</p> <pre>char letters[26]; *letters = 'a'; *letters = "a"; *letters[0] = 'a'; *(letters + 1) = 'a'; *letters + 1 = 'b';</pre>	*letters = 'a';

Total combine1		Study	<div>1/1</div>
<pre>int a[] = {6, 1, 9, 5, 1, 2, 3}; int x(0); for (auto e : a) x += e; cout << x << endl;</pre>	<div>Counts the elements in a</div> <div>Selects the largest value in a</div> <div>Has no effect</div> <div>Selects the smallest value in a</div> <div>Sums the elements in a</div>		
<div>[1414] What is the address of the first pixel in the last row of this image?</div> <div>Pixel *p; // address of pixel data</div> <div>int w, h; // width and height of image</div> <div>p + w + h</div> <div>p + w + (h - 1)</div> <div>p + w * h</div> <div>p + w * (h - 1)</div> <div>None of these are correct</div>	<div>p + w * (h - 1)</div>		
<div>[1415] Which returns the last pixel on the first row of this image?</div> <div>Pixel *p; // address of pixel data</div> <div>int w, h; // width and height of image</div> <div>*p + w - 1</div> <div>None of these are correct</div> <div>*(p + w) - 1</div> <div>p + w - 1</div> <div>*(p + w - 1)</div>	<div>*(p + w - 1)</div>		
<div>[1416] Which returns the last pixel on the first row of this image?</div> <div>Pixel *p; // address of pixel data</div> <div>int w, h; // width and height of image</div> <div>p[w - 1]</div> <div>*p[w - 1]</div> <div>None of these are correct</div> <div>p[w] - 1</div> <div>p + w - 1</div>	<div>p[w - 1]</div>		
<div>[1417] What is the equivalent array notation?</div> <div>int dates[10];</div> <div>cout << (*dates + 2) + 2 << endl;</div> <div>dates[0] + 4</div> <div>dates[2] + 2</div> <div>dates[2]</div> <div>dates[0] + 2</div> <div>&dates[2]</div>	<div>dates[0] + 4</div>		
<div>[1418] What is the equivalent array notation?</div> <div>int dates[10];</div> <div>cout << (dates + 2) << endl;</div> <div>dates[2] + 2</div> <div>&dates[2]</div> <div>dates[0] + 2</div> <div>dates[2]</div> <div>dates[0] + 4</div>	<div>&dates[2]</div>		
<div>[1419] What is the equivalent array notation?</div> <div>int dates[10];</div> <div>cout << *(dates + 2) << endl;</div> <div>dates[2] + 2</div> <div>dates[0] + 4</div> <div>dates[2]</div> <div>&dates[2]</div> <div>dates[0] + 2</div>	<div>dates[2]</div>		
<div>[1420] What is the equivalent array notation?</div> <div>int dates[10];</div> <div>cout << (*dates) + 2 << endl;</div> <div>&dates[2]</div> <div>dates[0] + 2</div> <div>dates[0] + 4</div> <div>dates[2]</div> <div>dates[2] + 2</div>	<div>dates[0] + 2</div>		



<pre>int dates[10]; cout << *dates + 2 << endl; &dates[2] dates[2] + 2 dates[0] + 4 dates[2] dates[0] + 2</pre>	
<p>[1422] What is the equivalent array notation?</p> <pre>int dates[10]; cout << *(dates + 2) + 2 << endl; &dates[2] dates[0] + 4 dates[0] + 2 dates[2] dates[2] + 2</pre>	<pre>dates[2] + 2</pre>
<p>[1423] What is the equivalent address-offset notation?</p> <pre>int a[] = {1, 2, 3, 4, 5, 6, 7}; int *p = a; cout << a[1] * 2 << endl; None of these <p>*p + 1 * 2 p + 1 * 2 (*p + 1) * 2 *(p + 1) * 2</p></pre>	<pre>*(p + 1) * 2</pre>
<p>[1424] What prints?</p> <pre>int a[] = {1, 3, 5, 7, 9}; int *p = a; cout << *p++; cout << *p << endl;</pre> <p>13 None of these 33 22 12</p>	<pre>13</pre>
<p>[1425] What prints?</p> <pre>int a[] = {1, 3, 5, 7, 9}; int *p = a; cout << *++p; cout << *p << endl;</pre> <p>33 13 None of these 22 12</p>	<pre>33</pre>
<p>[1426] What prints?</p> <pre>int a[] = {1, 3, 5, 7, 9}; int *p = a; cout << ++*p; cout << *p << endl;</pre> <p>13 12 None of these 22 33</p>	<pre>22</pre>
<p>[1427] Which pointer initialization is illegal?</p> <pre>int a[] = {1, 3, 5, 7, 9}; int *p3 = &a[1]; None of these int *p1 = a; int *p4 = &a; int *p2 = a + 3;</pre>	<pre>int *p4 = &a;</pre>

Total combine1		Study	<div>1/1</div>
<div>string countries[] = {"Andorra", "Albania", . . . };</div> <div>len(countries)</div> <div>countries.length</div> <div>sizeof(countries) * sizeof(countries[0])</div> <div>sizeof(countries)</div> <div>None of these</div>			
<div>[1429] Which expression returns the number of countries?</div> <div>string countries[] = {"Andorra", "Albania", . . . };</div> <div>sizeof(countries)</div> <div>len(countries)</div> <div>sizeof(countries) / sizeof(string)</div> <div>None of these</div> <div>sizeof(countries) * sizeof(countries[0])</div>		sizeof(countries) / sizeof(string)	
<div>[1430] Which expression returns the number of countries?</div> <div>string countries[] = {"Andorra", "Albania", . . . };</div> <div>len(countries)</div> <div>sizeof(countries) * sizeof(countries[0])</div> <div>sizeof(countries)</div> <div>None of these</div> <div>sizeof(countries) / sizeof(countries[0])</div>		sizeof(countries) / sizeof(countries[0])	
<div>[1431] Which array definition is illegal?</div> <div>int SIZE = 3;</div> <div>int a1[SIZE];</div> <div>int a2[3];</div> <div>int a3[3]{};</div> <div>int a4[] = {1, 2, 3};</div> <div>int a5[3] = {1, 2};</div> <div>a2</div> <div>a3</div> <div>None of these</div> <div>a1</div> <div>a5</div>		a1	
<div>[1432] Which array definition contains undefined values?</div> <div>int SIZE = 3;</div> <div>int a1[SIZE];</div> <div>int a2[3];</div> <div>int a3[3]{};</div> <div>int a4[] = {1, 2, 3};</div> <div>int a5[3] = {1, 2};</div> <div>a3</div> <div>a1</div> <div>None of these</div> <div>a5</div> <div>a2</div>		a2	
<div>[1433] Which array definition is initialized to all zeros?</div> <div>int SIZE = 3;</div> <div>int a1[SIZE];</div> <div>int a2[3];</div> <div>int a3[3]{};</div> <div>int a4[] = {1, 2, 3};</div> <div>int a5[3] = {1, 2};</div> <div>a5</div> <div>a2</div> <div>None of these</div> <div>a3</div> <div>a1</div>		a3	
<div>[1434] Which array definition produces {0, 1, 2}?</div> <div>int SIZE = 3;</div> <div>int a1[SIZE];</div> <div>int a2[3];</div> <div>int a3[3]{};</div> <div>int a4[] = {1, 2, 3};</div> <div>int a5[3] = {1, 2};</div> <div>a5</div> <div>a3</div> <div>None of these</div> <div>a2</div> <div>a1</div>		None of these	



<div><div>const int SIZE = 3; int a1[SIZE]; int a2[3]; int a3[3]{}; int a4[] = {1, 2, 3}; int a5[2] = {1, 2, 3}; a2 a5 a3 None of these a1</div></div>	<div><div></div></div>
<div><div>[1436] Which array definition produces {1, 2, 0}?</div><div><div>int SIZE = 3; int a1[SIZE]; int a2[3]; int a3[3]{}; int a4[] = {1, 2, 3}; int a5[3] = {1, 2}; a3 a5 a2 a1 None of these</div></div></div>	<div><div>a5</div></div>
<div><div>An incomplete type and a forward reference generally mean the same thing. In C++ using == to compare one array to another is permitted (if meaningless). You must use an integral constant or literal to specify the size of a built-in C++ array. The reinterpret_cast instruction changes way that a pointer's indirect value is interpreted. If p is a pointer to a structure, and the structure contains a data member x, you can access the data member by using the notation: (*p).x C++ arrays have no support for bound-checking. In C++ assigning one array to another is illegal The allocated size of a built-in C++ array cannot be changed during runtime. The size of the array is not stored along with its elements. If img is a pointer to the first byte in an image loaded into memory, Pixel is a structure as defined in your textbook, you can create a Pixel pointer pointing to the image by writing: Pixel p = reinterpret_cast<Pixel >(img); The subscripts of a C++ array range from 0 to the array size - 1. C++ arrays have no built-in functions for inserting and deleting. A forward reference can be used when you want to use a pointer to a structure as a data member without first defining the entire structure. The elements of a C++ array created in a function are allocated on the stack. The elements of a C++ array created outside of a function are allocated in the static-storage area. The elements of a C++ string array with no explicit initialization, created in a function will be set to the empty string. Explicitly initializing an array like this: int a[3] = {1, 2, 3}; requires the size to be the same or larger than the number of elements supplied. In C++ printing an array name prints the address of the first element in the array. In C++ there is no separate array variable. The array name is a symbolic representation of the address of the first element in the array. In C++ initializing an array with the contents of another is illegal. C++ arrays produce undefined results if you access an element outside the array. Explicitly initializing an array like this: int a[] = {1, 2, 3}; works in all versions of C++.</div></div>	<div><div>True</div></div>



You may use any kind of integral variable to specify the size of a built-in C++ array.

The elements of a C++ string array with no explicit initialization, created in a function will be set to null.

Explicitly initializing an array like this: `int a[3] = {1, 2, 3};` requires the size to be the same or smaller than the number of elements supplied.

In C++ using `==` to compare one array to another is illegal.

The allocated size of a built-in C++ array may be changed during runtime

If `img` is a pointer to the first byte in an image loaded into memory, `Pixel` is a structure as defined in your textbook, you can create a `Pixel` pointer pointing to the image by writing:
`Pixel p = static_cast<Pixel>(img);`

The `reinterpret_cast` instruction produces a temporary value by converting its argument.

In C++ initializing an array with the contents of another is permitted.

C++ arrays use bound-checking when you access their elements with the `at()` member function.

The elements of a C++ array created in a function are allocated on the heap.

In C++ assigning one array to another is permitted.

C++ arrays throw an `out_of_bounds` exception if you access an element outside the array.

In C++ an array variable and the array elements are separate. The array variable contains the address of the first element in the array.

In C++ printing an array name prints the value of the first element in the array.

The elements of a C++ int array with no explicit initialization, created in a function will be set to zero.

C++ arrays can be allocated with a size of 0.

The `static_cast` instruction changes way that a pointer's indirect value is interpreted.

The size of the array is stored along with its elements.

The allocated size of a built-in C++ array may be changed during runtime

A forward reference can be used when you want to use a structure as a data member without first defining the entire structure.

The elements of a C++ array created outside of a function are allocated on the stack.

If `p` is a pointer to a structure, and the structure contains a data member `x`, you can access the data member by using the notation: `*p->x`

C++ arrays offer built-in member functions for inserting and deleting.

Explicitly initializing an array like this: `int a[] = {1, 2, 3};` only works in C++ 11.

[1501] Below is a cumulative algorithm using an array and a range-based loop. What is printed? (Assume this is inside `main()` with all includes, etc.)

```
int a[] = {2, 4, 6, 8};
int sum = 0;
for (auto e : a) sum += e;
cout << "sum->" << sum << endl;
```

Compiles but crashes with an endless loop.
Does not compile. Cannot use range-loop on arrays.
sum->20
sum->0
Compiles and runs, but results are undefined.

sum->20

[1502] Below is a cumulative algorithm using an array and a range-based loop. What is printed? (Assume this is inside `main()` with all includes, etc.)

```
int a[] = {2, 4, 6, 8};
int sum;
for (auto e : a) sum += e;
cout << "sum->" << sum << endl;
```

Compiles and runs, but results are undefined.
sum->20
sum->8
Does not compile. Cannot use range-loop on arrays.
Compiles but crashes with an endless loop.

Compiles and runs, but results are undefined.



<pre>int a[] = {2, 4, 6, 8}; int sum = 0; for (auto e : a) sum += e; cout << "sum->" << e << endl;</pre> <p>Does not compile; e is undefined. Does not compile. Cannot use range-loop on arrays. Compiles and runs, but results are undefined. sum->20 sum->8</p>	
<p>[1504] Below is a cumulative algorithm using an array and a range-based loop. What is printed? (Assume this is inside main() with all includes, etc.)</p> <pre>int a[] = {2, 4, 6, 8}; int sum = 0; for (auto e : a) sum += e; cout << "sum->" << sum << endl;</pre> <p>Does not compile. Cannot use range-loop on arrays. sum->8 Compiles and runs, but results are undefined. sum->20 Does not compile; e is undefined.</p>	sum->8
<p>[1505] Below is a cumulative algorithm using an array and an iterator-based loop. What is printed? (Assume all includes have been added, etc.)</p> <pre>double average(const int *beg, const int *end) { double sum = 0; size_t count = end - beg; while (beg != end) sum += *beg++; return sum / count; } int main() { int a[] = {2, 4, 6, 8}; cout << average(begin(a), end(a)) << endl; }</pre> <p>4 5 Does not compile 6 Endless loop when run; likely crashes.</p>	5
<p>[1506] Below is a cumulative algorithm using an array and an iterator-based loop. What is printed? (Assume all includes have been added, etc.)</p> <pre>double average(const int beg, const int end) { double sum = 0; size_t count = end - beg; while (beg != end) sum += *beg++; return sum / count; } int main() { int a[] = {2, 4, 6, 8}; cout << average(begin(a), end(a) - 1) << endl; }</pre> <p>Endless loop when run; likely crashes. Does not compile 4 5 6</p>	4



<pre>double average(const int beg, const int end) { double sum = 0; size_t count = end - beg; while (beg != end) sum += *beg++; return sum / count; } int main() { int a[] = {2, 4, 6, 8}; cout << average(begin(a) + 1, end(a)) << endl; }</pre> <p>6 4 5 Does not compile Endless loop when run; likely crashes.</p>	
<p>[1508] Below is a cumulative algorithm using an array and an iterator-based loop. What is printed? (Assume all includes have been added, etc.)</p> <pre>double average(const int beg, const int end) { double sum = 0; size_t count = end - beg; while (beg != end) sum += *beg++; return sum / count; } int main() { int a[] = {2, 4, 6, 8}; cout << average(end(a), begin(a)) << endl; }</pre> <p>Does not compile Endless loop when run; likely crashes. 5 6 4</p>	<p>Endless loop when run; likely crashes.</p>
<p>[1509] Below is a cumulative algorithm using an array and an iterator-based loop. What is printed? (Assume all includes have been added, etc.)</p> <pre>double average(const int beg, const int end) { if (end <= beg) return 0.0 / 0.0; // nan double sum = 0; size_t count = end - beg; while (beg != end) sum += *beg++; return sum / count; } int main() { int a[] = {2, 4, 6, 8}; cout << average(end(a), begin(a)) << endl; }</pre> <p>4 Does not compile 5 Not a number (NaN) Endless loop when run; likely crashes.</p>	<p>Not a number (NaN)</p>

Total combine1		Study	
<pre>double average(const int beg, const int end) { double sum = 0; size_t count = end - beg; while (beg != end) sum += *beg++; return sum / count; } int main() { int a[] = {2, 4, 6, 8}; cout << average(a, a + 1) << endl; }</pre> <p>Does not compile</p> <p>3 2 5 4</p>			
<p>[1511] Below is a cumulative algorithm using an array and an iterator-based loop. What is printed? (Assume all includes have been added, etc.)</p> <pre>double average(const int beg, const int end) { double sum = 0; size_t count = end - beg; while (beg != end) sum += *beg++; return sum / count; } int main() { int a[] = {2, 4, 6, 8}; cout << average(a, a + 2) << endl; }</pre> <p>Does not compile</p> <p>5 3 4 2</p>	3		
<p>[1512] Below is a cumulative algorithm using an array and an iterator-based loop. What is printed? (Assume all includes have been added, etc.)</p> <pre>double average(const int beg, const int end) { double sum = 0; size_t count = end - beg; while (beg != end) sum += *beg++; return sum / count; } int main() { int a[] = {2, 4, 6, 8}; cout << average(a + 1, a + 3) << endl; }</pre> <p>5 2 Does not compile 4 3</p>	5		
<p>[1513] Below is a cumulative algorithm using an array and an iterator-based loop. What is printed? (Assume all includes have been added, etc.)</p> <pre>double average(const int beg, const int end) { double sum = 0; size_t count = end - beg; while (beg != end) sum += *beg++; return sum / count; } int main() { int a[] = {2, 4, 6, 8}; cout << average(a, a + 3) << endl; }</pre> <p>5 Does not compile 4 2 3</p>	4		

Total combine1		Study	
<pre>const int a[] = {2, 4, 6, 8}; cout << mystery(a, 4) << endl; void mystery(const int a[], size_t n); int mystery(int a[], size_t n); int mystery(const int a*, size_t n); int mystery(const int *a, size_t n); int mystery(const int[] a, size_t n);</pre>			
<p>[1515] What is the correct prototype for mystery? (It may modify the array.)</p> <pre>const int a[] = {2, 4, 6, 8}; cout << mystery(a, 4) << endl; int mystery(int[] a, size_t n); int mystery(int a, size_t n); int mystery(int *a, size_t n); int mystery(int a*, size_t n); void mystery(const int a[], size_t n);</pre>	<pre>int mystery(int *a, size_t n);</pre>		
<p>[1516] What is printed here? (Assume all includes have been added. Assume 4-bytes per int, 8 bytes per pointer.)</p> <pre>size_t len(const int a[]) { return sizeof(a) / sizeof(a[0]); } int main() { int a[] = {2, 4, 6, 8}; cout << len(a) << endl; } 2 Does not compile 1 4</pre>	2		
<p>[1517] What is printed here? (Assume all includes have been added. Assume 4-bytes per int, 8 bytes per pointer.)</p> <pre>int main() { int a[] = {2, 4, 6, 8}; cout << sizeof(a) / sizeof(a[0]) << endl; } Does not compile 4 1 2</pre>	4		
<p>[1518] What is printed here? (Assume all includes have been added. Assume 4-bytes per int, 8 bytes per pointer.)</p> <pre>size_t len(const int a, const int b) { return b - a; } int main() { int a[] = {2, 4, 6, 8}; cout << len(begin(a), end(a)) << endl; } Does not compile 4 2 1</pre>	4		
<p>[1519] What is printed here? (Assume all includes have been added. Assume 4-bytes per int, 8 bytes per pointer.)</p> <pre>size_t len(const int a, const int b) { return b - a; } int main() { int a[] = {2, 4, 6, 8}; cout << len(a, a + 3) << endl; } 2 3 4 Does not compile</pre>	3		



<pre>int odds(int a[], size_t len) { int sum = 0; for (size_t i = 0; i < len; i++) if (a[i] % 2 == 1) sum += a[i]++; return sum; } int main() { int a[] = {1, 3, 5}; cout << odds(a, 3) << odds(a, 2) << odds(a, 1) << endl; } 999 900 300 941 Does not compile</pre>	
<p>[1521] What does this function do?</p> <pre>int mystery(const int a[], size_t n) { int x = n - 1; while (n > 0) { n--; if (a[n] > a[x]) x = n; } return x; }</pre> <p>Returns the largest number in the array Returns the index of the last occurrence of the largest number in the array Returns the smallest number in the array Returns the index of the first occurrence of the largest number in the array Does not compile</p>	<p>Returns the index of the last occurrence of the largest number in the array</p>
<p>[1522] What does this function do?</p> <pre>int mystery(const int a[], size_t n) { int x = n - 1; while (n > 0) { n--; if (a[n] < a[x]) x = n; } return x; }</pre> <p>Returns the smallest number in the array Returns the index of the last occurrence of the smallest number in the array Does not compile Returns the index of the first occurrence of the smallest number in the array Returns the largest number in the array</p>	<p>Returns the index of the last occurrence of the smallest number in the array</p>
<p>[1523] What does this function do?</p> <pre>int mystery(const int a[], size_t n) { int x = a[n - 1]; while (n > 0) { n--; if (a[n] < a[x]) x = a[n]; } return x; }</pre> <p>Returns the index of the first occurrence of the smallest number in the array Returns the largest number in the array Returns the index of the last occurrence of the smallest number in the array Returns the smallest number in the array Does not compile</p>	<p>Returns the smallest number in the array</p>



<pre>int mystery(const int a[], size_t n) { int x = a[n - 1]; while (n > 0) { n--; if (a[n] > a[x]) x = a[n]; } return x; }</pre> <p>Returns the index of the last occurrence of the smallest number in the array</p> <p>Does not compile</p> <p>Returns the largest number in the array</p> <p>Returns the smallest number in the array</p> <p>Returns the index of the first occurrence of the smallest number in the array</p>	
<p>[1525] What is printed?</p> <pre>int mystery(const int a[], size_t n) { int x = a[n - 1]; while (n > 0) { n--; if (a[n] > a[x]) x = a[n]; } return x; }</pre> <pre>int main() { int a[] = {1, 3, 5, 3, 5, 4}; cout << mystery(a, 6) << endl; }</pre>	5
<p>[1526] What is printed?</p> <pre>int mystery(const int a[], size_t n) { int x = n - 1; while (n > 0) { n--; if (a[n] < a[x]) x = n; } return x; }</pre> <pre>int main() { int a[] = {1, 2, 5, 2, 5, 4}; cout << mystery(a, 6) << endl; }</pre> <p>1</p> <p>2</p> <p>3</p> <p>None of these</p> <p>4</p>	None of these
<p>[1527] What is printed?</p> <pre>int mystery(const int a[], size_t n) { int x = n - 1; while (n > 0) { n--; if (a[n] < a[x]) x = n; } return x; }</pre> <pre>int main() { int a[] = {4, 2, 5, 2, 5, 4}; cout << mystery(a, 6) << endl; }</pre> <p>1</p> <p>4</p> <p>None of these</p> <p>3</p> <p>2</p>	3



<pre>int mystery(const int a[], size_t n) { int x = n - 1; while (n > 0) { n--; if (a[n] > a[x]) x = n; } return x; }</pre> <p>int main()</p> <pre>{ int a[] = {4, 2, 5, 2, 5, 4}; cout << mystery(a, 6) << endl; }</pre> <p>None of these</p> <p>4</p> <p>3</p> <p>1</p> <p>2</p>	
<p>[1529] What is printed?</p> <pre>int mystery(const int a[], size_t n) { int x = 0; for (size_t i = 0; i < n; i++) if (a[i] > a[x]) x = i; return x; }</pre> <p>int main()</p> <pre>{ int a[] = {4, 2, 5, 2, 5, 4}; cout << mystery(a, 6) << endl; }</pre> <p>5</p> <p>None of these</p> <p>0</p> <p>2</p> <p>4</p>	2
<p>[1530] What is printed?</p> <pre>int mystery(const int a[], size_t n) { int x = 0; for (size_t i = 0; i < n; i++) if (a[i] < a[x]) x = i; return x; }</pre> <p>int main()</p> <pre>{ int a[] = {4, 2, 5, 2, 5, 4}; cout << mystery(a, 6) << endl; }</pre> <p>None of these</p> <p>2</p> <p>0</p> <p>1</p> <p>3</p>	1
<p>[1531] What is printed?</p> <pre>const int mystery(const int p, size_t n) { const int x = p, y = p + n; while (++p != y) { if (p > x) x = p; } return x; }</pre> <p>int main()</p> <pre>{ int a[] = {1, 2, 3, 4, 5, 1}; cout << *(mystery(a, 6)) << endl; }</pre> <p>0</p> <p>5</p> <p>2</p> <p>None of these</p> <p>4</p>	5



<pre>const int mystery(const int p, size_t n) { const int x = p, y = p + n; while (++p != y) { if (p > x) x = p; } return x; } int main() { int a[] = {1, 2, 3, 4, 5, 1}; cout << *(mystery(a, 6)) << endl; } 4 5 2 None of these 0</pre>	
<p>[1533] What does this function do?</p> <pre>double mystery(const double a[], size_t len) { double x = a[0]; for (size_t i = 1; i < len; i++) if (a[i] > x) x = a[i]; return x; }</pre> <p>Does not compile Returns the largest number in the array Returns the smallest number in the array Undefined. Depends on the input.</p>	<p>Returns the largest number in the array</p>
<p>[1534] What does this function do?</p> <pre>double mystery(const double a[], size_t len) { double x = a[0]; for (size_t i = 1; i < len; i++) if (a[i] < x) x = a[i]; return x; }</pre> <p>Returns the largest number in the array Does not compile Returns the smallest number in the array Undefined. Depends on the input.</p>	<p>Returns the smallest number in the array</p>
<p>[1535] What does this function do?</p> <pre>double mystery(const double a[], size_t len) { double x = 0; for (size_t i = 0; i < len; i++) if (a[i] > x) x = a[i]; return x; }</pre> <p>Undefined. Depends on the input. Does not compile Returns the largest number in the array Returns the smallest number in the array</p>	<p>Undefined. Depends on the input.</p>
<p>[1536] What does this function do?</p> <pre>double mystery(const double a[], size_t len) { double x = 0; for (size_t i = 0; i < len; i++) if (a[i] < x) x = a[i]; return x; }</pre> <p>Returns the largest number in the array Returns the smallest number in the array Undefined. Depends on the input. Does not compile</p>	<p>Undefined. Depends on the input.</p>

Total combine1		Study	...
<pre>template <typename T> ostream& mystery(ostream& out, const T* p, size_t n) { out << '['; if (n) { out << p[0]; for (size_t i = 1; i < n; i++) out << ", " << p[i]; } out << "]; return out; }</pre> <p>A cumulative algorithm An extreme values algorithm An iterator algorithm None of these A fencepost algorithm</p>			
<p>[1538] What is printed?</p> <pre>template <typename T> ostream& mystery(ostream& out, const T* p, size_t n) { out << '['; if (n) { out << p[0]; for (size_t i = 1; i < n; i++) out << ", " << p[i]; } out << "]; return out; }</pre> <p>int a[] = {1,2,3,4,5,1}; mystery(cout, a, 4) << endl;</p> <p>[1, 2, 3] [1, 2, 3, 4, 5, 1] None of these or undefined output. [1, 2, 3, 4, 5] [1, 2, 3, 4]</p>		<p>[1, 2, 3, 4]</p>	
<p>[1539] What is printed?</p> <pre>template <typename T> ostream& mystery(ostream& out, const T* p, size_t n) { out << '['; if (n) { out << p[0]; for (size_t i = 1; i < n; i++) out << ", " << p[i]; } out << "]; return out; }</pre> <p>int a[] = {1,2,3,4,5,1}; mystery(cout, a, sizeof(a)) << endl;</p> <p>[1, 2, 3, 4, 5, 1] [1, 2, 3, 4] [1, 2, 3, 4, 5] None of these or undefined output. [1, 2, 3]</p>		<p>None of these or undefined output.</p>	
<p>[1540] What is printed?</p> <pre>template <typename T> ostream& mystery(ostream& out, const T* p, size_t n) { out << '['; if (n) { out << p[0]; for (size_t i = 1; i < n; i++) out << ", " << p[i]; } out << "]; return out; }</pre> <p>... int a[] = {1,2,3,4,5,1};</p> <p>mystery(cout, a, sizeof(a) / sizeof(a[0])) << endl;</p> <p>None of these or undefined output. [1, 2, 3, 4] [1, 2, 3] [1, 2, 3, 4, 5] [1, 2, 3, 4, 5, 1]</p>		<p>[1, 2, 3, 4, 5, 1]</p>	



```
template <typename T>
ostream& mystery(ostream& out, const T* p, size_t n)
{
    out << '[';
    if (n) {
        out << p[0];
        for (size_t i = 1; i < n; i++)
            out << ", " << p[i];
        out << "];"
    }
    return out;
}

...
int a[] = {1,2,3,4,5,1};

mystery(cout, a, 0) << endl;
[0]
Does not compile. Arrays cannot be 0 length.
[]
[1]
No output
```

Elements always allocated on the heap

How arrays are passed to functions

What happens to an array when passed to a function

const int *array

int * const array

const int * const array

sizeof(a) / sizeof(a[0])

end(a) - begin(a)

for (auto e : a) ..

x = 0; for (auto e : a) x += e;

x = a[0]; for (auto e: a) if (e > x) x = e;

auto p = a; while (p != end(a)) p++;

cout << a[0]; while (i < len) cout << ", " << a[i++];

vector

by address

decays

Elements may not be modified; pointer may be

Elements in may be modified; pointer may not

Neither pointer nor elements in may be modified

Elements in array using arithmetic

Elements in array using pointer difference

A range-based loop

Cumulative algorithm

Extreme values algorithm

Iterator-based loop

Fence-post algorithm



An array passed to a function decays to a pointer.

An array passed to a function `f(int * const a, ...)` may have its elements changed.

The elements of an array may be allocated on the stack.

If `p` points to the first element in `[1, 3, 5]` then `cout << **p` prints 2.

The library function `begin(a)` returns a pointer to the first element in the array `a`.

The elements of an array may be allocated in the static storage area.

Arrays generally have higher performance than a vector.

The function `mystery(const int, const int)` likely employs an iterator loop.

The expression `begin(a) + 1` returns a pointer to the second element in the array `a`.

Array subscripts are not range checked

An array passed to a function is passed by address.

If `size_t len = 0`; then `len - 1` is the largest possible unsigned number.

If `p` points to the first element in `[1, 3, 5]` then `cout << *++p` prints 3.

The algorithm that finds the address of the smallest element in an array is called an extreme values algorithm.

The expression **`p++` means the same as** `(p++)`.

Before passing an array to a function, `sizeof(a)/sizeof(a[0])` will tell the number of elements in the array.

For systems programming (such as operating systems), arrays are used more often than vectors.

The library function `end(a)` returns a pointer to position right past the last element in the array `a`.

For embedded systems, arrays are preferred over vector.

The parameter declarations `int *p` and `int p[]` mean the same thing.

The algorithm that prints elements separated by commas is called the fencepost algorithm.

The elements of a vector are allocated on the heap.

A vector variable may be allocated on the stack.

Before passing an array to a function, `sizeof(a)` will tell you the array's allocated size, but not the number of elements.

Total combine1		Study	<div><div></div><div></div><div></div></div>
<p>After passing an array to a function, sizeof(a)/sizeof(a[0]) will tell the number of elements in the array.</p> <p>If p points to the first element in [1, 3, 5] then cout << *++p prints 1.</p> <p>If p points to the first element in [1, 3, 5] then cout << ++*p prints 1.</p> <p>The library function begin(a) returns a pointer to the element right before the first in the array a.</p> <p>For embedded systems, vector is preferred over arrays.</p> <p>For systems programming (such as operating systems), vectors are used more often than arrays.</p> <p>For an equivalent number of elements, a vector will use less memory than an array.</p> <p>The expression p++ means the same as (p)++.</p> <p>An array passed to a function f(const int *a, ...) may have its elements changed.</p> <p>The elements of a vector may be allocated on the stack.</p> <p>For an equivalent number of elements, a vector will use more memory than an array.</p> <p>The algorithm that prints elements separated by commas is called a cumulative algorithm.</p> <p>The algorithm that finds the position of the largest element in an array is called a cumulative algorithm.</p> <p>The algorithm that finds the position of the largest element in an array is called a cumulative algorithm.</p> <p>After passing an array to a function, sizeof(a) will tell you the array's allocated size, but not the number of elements.</p> <p>If p points to the first element in [1, 3, 5] then cout << *p++ prints 3.</p> <p>The library function end(a) returns a pointer to the last element in the array a.</p> <p>A vector generally has higher performance than an array.</p> <p>If size_t len = 0; then len - 1 is the smallest possible unsigned number.</p> <p>The expression begin(a) + 1 returns a pointer to the first element in the array a.</p> <p>The function mystery(const int, const int) likely employs a counter-controlled loop.</p> <p>An array passed to a function is passed by reference.</p>			
<p>[1601] Below is a partially-filled array. If you are adding elements to this array in a loop, what is the correct loop bounds condition?</p> <pre>const size_t MAX = 100; double nums[MAX]; size_t size = 0; while (MAX < size) ... while (size < MAX) ... while (size <= MAX) ... for (size = 0; size < MAX; size++) ...</pre>		while (size < MAX) ...	
<p>[1602] Below is a partially-filled array. When adding elements to this array in a loop, what statement(s) correctly updates the array with value?</p> <pre>const size_t MAX = 100; double nums[MAX]; size_t size = 0; double value; nums[size] = value; nums[size++] = value; nums[++size] = value; size++; nums[size] = value;</pre>		nums[size++] = value;	
<p>[1603] Below is a partially-filled array. If you have a sentinel loop where the sentinel is a negative number, which of these conditions correctly reads the number named value?</p> <pre>const size_t MAX = 100; double nums[MAX]; size_t size = 0; double value; cin >> value; if (value < 0) break; if (! (cin >> value) value < 0) break; cin >> value; if (cin.fail() && value < 0) break; if (value >= 0 && cin >> value) ... // process value</pre>		if (! (cin >> value) value < 0) break;	

Total combine1		Study	<div>1/1</div>
<pre>const size_t MAX = 100; double nums[MAX]; size_t size = 0; double& back(double a[], size_t size); double& back(double a[], size_t& size); double& back(const double a[], size_t& size); double& back(double a[], size_t size, size_t MAX);</pre>			
<p>[1605] Below is a declaration for a partially-filled array. What is the correct prototype for a function add() that appends a new element to the end of the array and returns true if successful?</p> <pre>const size_t MAX = 100; double nums[MAX]; size_t size = 0; bool add(double a[], size_t MAX, double e); bool add(double a[], size_t& size, double e); bool add(double a[], size_t size, size_t MAX, double e); bool add(double a[], size_t& size, size_t MAX, double e);</pre>	<pre>bool add(double a[], size_t& size, size_t MAX, double e);</pre>		
<p>[1606] Below is a declaration for a partially-filled array. What is the correct prototype for a function insert() that inserts a new element at position pos in the array, shifts the remaining elements right, and returns true if successful?</p> <pre>const size_t MAX = 100; double nums[MAX]; size_t size = 0; bool insert(double a[], size_t& size, double e, size_t pos); bool insert(double a[], size_t MAX, double e, size_t pos); bool insert(double a[], size_t size, size_t MAX, double e, size_t pos); bool insert(double a[], size_t& size, size_t MAX, double e, size_t pos);</pre>	<pre>bool insert(double a[], size_t& size, size_t MAX, double e, size_t pos);</pre>		
<p>[1607] Below is a declaration for a partially-filled array. What is the correct prototype for a function delete() that deletes the element at position pos in the array, shifts the remaining elements left, and returns true if successful?</p> <pre>const size_t MAX = 100; double nums[MAX]; size_t size = 0; bool delete(double a[], size_t size, size_t pos); bool delete(double a[], size_t& size, size_t pos); bool delete(double a[], size_t MAX, size_t& pos); bool delete(const double a[], size_t& size, size_t pos);</pre>	<pre>bool delete(double a[], size_t& size, size_t pos);</pre>		
<p>[1608] Below is a mystery() function with no types for its parameter. What does the function do?</p> <pre>void mystery(a, b&, c, d, e) { b = 0; while (in >> n && b < c) a[b++] = n; }</pre> <p>Inserts input into a partially-filled array Deletes elements from a partially-filled array Appends input to the end of a partially-filled array.</p>	<p>Appends input to the end of a partially-filled array.</p>		



<pre>void mystery(a, b&, c, d, e) { for (i = d; i < b; i++) a[i] = a[i + 1]; b--; }</pre> <p>Inserts input into a partially-filled array Deletes elements from a partially-filled array Appends input to the end of a partially-filled array.</p>	
<p>[l6l0] Below is a mystery() function with no types for its parameter. What does the function do?</p> <pre>void mystery(a, b&, c, d, e) { for (i = b; i > d; i--) a[i] = a[i - 1]; a[d] = e; b++; }</pre> <p>Inserts input into a partially-filled array Deletes elements from a partially-filled array Appends input to the end of a partially-filled array.</p>	<p>Inserts input into a partially-filled array</p>
<p>[l6l1] Below is a template function, push(), that adds elements to the end of a partially-filled array, returning true if successful. The function has an error; what is the error?</p> <pre>template <typename T> bool push(T* a, size_t& size, size_t MAX, T e) { if (size < MAX) { a[size] = e; return true; } return false; }</pre> <p>a should be a const T* size should be incremented size should be passed by value Condition should be size <= MAX</p>	<p>size should be incremented</p>
<p>[l6l2] Below is pop(), a template function that works with a partially-filled array. The function copies the last element in the array into the output parameter e and returns true if successful; it returns false otherwise. What is the error?</p> <pre>template <typename T> bool pop(T* a, size_t& size, T& e) { if (size) { e = a[size]; size--; return true; } return false; }</pre> <p>a should be a const T* Condition should be !size size should be incremented The wrong value is assigned to e</p>	<p>The wrong value is assigned to e</p>
<p>[l6l3] Below is index(), a template function that works with a partially-filled array. The function searches the array a for the value e and returns its position. It returns NOT_FOUND if the value does not it exist in the array. The function contains an error; what is the error?</p> <pre>const size_t NOT_FOUND = static_cast<size_t>(-1); template <typename T> size_t index(const T* a, size_t& size, T e) { for (size_t i = 0; i < size; i++) if (a[i] == e) return i; return NOT_FOUND; }</pre> <p>a should not be a const T* e should be passed by reference The condition should go to i <= size size should not be passed by reference</p>	<p>size should not be passed by reference</p>



removed. The function contains an error; what is the error?

```
template <typename T>
int remove(T* a, size_t& size, T e)
{
    int removed = 0;
    size_t i = 0;
    while (i < size)
    {
        if (a[i] == e)
        {
            removed++;
            size--;
            for (size_t j = i; j < size; j++)
                a[j] = a[j + 1];
            i++;
        }
    }
    return removed;
}
```

- a should be a const T*
- size should not be passed by reference
- The condition should go to while (i <= size)
- Not all copies of e are necessarily removed

[1615] Below is insert(), a template function that works with a partially-filled array. The function inserts the argument e into the array, in sorted order. The function returns true if it succeeds, false otherwise. The function contains an error; what is the error?

```
template <typename T>
bool insert(T* a, size_t& size, size_t MAX, T e)
{
    if (size < MAX) return false;
    size_t i = 0;
    while (i < size)
    {
        if (a[i] > e) break;
        i++;
    }
    for (j = size; j > i; j--)
        a[j] = a[j - 1];
    a[i] = e;
    size++;

    return true;
}
```

The value is inserted into the wrong position

The second loop should start at i and go up to size

When a value is inserted, it erases one of the existing values

If there is room to insert, the function returns false instead of true

If there is room to insert, the function returns false instead of true

[1616] Below is insert(), a template function that works with a partially-filled array. The function inserts the argument e into the array, in sorted order. The function returns true if it succeeds, false otherwise. The function contains an error; what is the error?

```
template <typename T>
bool insert(T* a, size_t& size, size_t MAX, T e)
{
    if (size < MAX) return false;
    size_t i = 0;
    while (i < size)
    {
        if (a[i] > e) break;
        i++;
    }
    for (j = size; j > i; j--)
        a[j] = a[j - 1];
    a[i] = e;
    size++;

    return true;
}
```

The value is inserted into the wrong position

The second loop should start at i and go up to size

When a value is inserted, it erases one of the existing values

If the array is full, the function overwrites memory outside the array

If the array is full, the function overwrites memory outside the array.

Total combine1		Study	<div>1/1</div>
<p>true if it succeeds, false otherwise. The function contains an error; what is the error?</p> <pre>template <typename T> bool insert(T* a, size_t& size, size_t MAX, T e) { if (size >= MAX) return false; size_t i = 0; while (i < size) { if (a[i] > e) break; i++; } for (j = size; j > i; j--) a[j] = a[j - 1]; a[i] = e; return true; }</pre> <p>The value is inserted into the wrong position</p> <p>The second loop should start at i and go up to size</p> <p>Every time the function is called, an array element is "lost"</p> <p>The function writes over memory outside the array when it should not</p>			
<p>[1618] Which loop is used when inserting an element into an array?</p> <p>for (j = pos; j < size; j++) a[j] = a[j + 1]; for (j = size; j > pos; j--) a[j] = a[j - 1]; for (j = MAX; j > size; j--) a[j - 1] = a[j]; for (j = size; j < MAX; j++) a[j - 1] = a[j];</p>		for (j = size; j > pos; j--) a[j] = a[j - 1];	
<p>[1619] Which loop is used when deleting an element from an array?</p> <p>for (j = MAX; j > size; j--) a[j - 1] = a[j]; for (j = pos; j < size; j++) a[j] = a[j + 1]; for (j = size; j > pos; j--) a[j] = a[j - 1]; for (j = size; j < MAX; j++) a[j - 1] = a[j];</p>		for (j = pos; j < size; j++) a[j] = a[j + 1];	
<p>[1620] Assume you have a partially filled array a, with variables size and MAX (capacity). To append value to the array, which of these assignments is correct?</p> <p>a[size] = value; a[size + 1] = value; a[size - 1] = value; a[MAX - 1] = value;</p>		a[size] = value;	
<p>[1621] Below is startsWith(), a template function that works with two partially-filled arrays. The function returns true if the array a "starts with" the same elements as the array b, false otherwise. The function contains an error; what is the error?</p> <pre>template <typename T> bool startsWith(const T* a, size_t sizeA, const T* b, size_t sizeB) { if (sizeA > sizeB) return false; for (size_t i = 0; i < sizeB; i++) if (a[i] != b[i]) return false; return true; }</pre> <p>The condition i < sizeB should be i <= sizeB The condition a[i] != b[i] should be b[i] == a[i] sizeA and sizeB should both be passed by reference The condition (sizeA > sizeB) should be (sizeB > sizeA)</p>		The condition (sizeA > sizeB) should be (sizeB > sizeA)	
<p>[1622] Below is endsWith(), a template function that works with two partially-filled arrays. The function returns true if the array a "ends with" the same elements as the array b, false otherwise. The function contains an error; what is the error?</p> <pre>template <typename T> bool endsWith(T* a, size_t sizeA, T* b, size_t sizeB) { if (sizeA < sizeB) return false; size_t diff = sizeA - sizeB; for (size_t i = 0; i < sizeB; i++) if (a[i + diff] != b[i]) return false; return true; }</pre> <p>The arrays a and b should be const T* sizeA and sizeB should both be passed by reference The condition (sizeA < sizeB) should be (sizeA > sizeB) The condition a[i + diff] != b[i] should be a[i - diff] == b[i]</p>		The arrays a and b should be const T*	



function contains an error; what is the error?

```
template <typename T>
int removeDupes(T* a, size_t& size)
{
    int count = 0;
    for (size_t i = 0; i < size; i++) {
        for (size_t j = i + 1; j < size; j++) {
            if (a[i] == a[j]) { // duplicate
                size--; count++;
            }
        }
    }
    return count;
}
```

The array parameter should be const T

It removes some duplicates, but not all of them

It returns a different number than the actual elements removed

It produces undefined behavior by exceeding the bounds of the array

In a partially-filled array, the capacity may be less than the array's size.

When inserting a value into a partially-filled array, in ascending order, the insertion position may be the same as capacity.

When inserting elements into a partially-filled array, the array should be declared const.

When comparing two partially-filled arrays for equality, both arrays should not be declared const.

When deleting an element from a partially-filled array, it is an error if the index of the element to be removed is < size.

When inserting a value into a partially-filled array, elements following the insertion position are shifted to the left.

In a partially-filled array, the size represents the allocated size of the array.

In a partially-filled array, the capacity represents the effective size of the array.

In a partially-filled array, all of the elements are not required to contain meaningful values

When inserting an element into a partially-filled array, it is an error if size < capacity.

In a partially-filled array, all of the elements contain meaningful values

When deleting elements from a partially-filled array, the array should be declared const.

In a partially-filled array capacity represents the number of elements that are in use.

When searching for the index of a particular value in a partially-filled array, the array should not be declared const.

When inserting a value into a partially-filled array, in ascending order, the insertion position is the index of the first value smaller than the value.

False



<p>When inserting a value into a partially-filled array, in ascending order, the insertion position may be the same as size.</p> <p>When inserting a value into a partially-filled array, in descending order, the insertion position is the index of the first value smaller than the value.</p> <p>When removing an element from a partially-filled array, elements following the deletion position are shifted to the left.</p> <p>When deleting elements from a partially-filled array, the array should not be declared const.</p> <p>In a partially-filled array size represents the number of elements that are in use.</p> <p>When inserting a value into a partially-filled array, elements following the insertion position are shifted to the right.</p> <p>In a partially-filled array, the capacity represents the allocated size of the array.</p> <p>When searching for the index of a particular value in a partially-filled array, the array should be declared const.</p> <p>When inserting an element into a partially-filled array, it is an error if size >= capacity.</p> <p>In a partially-filled array, the size may be less than the array's capacity.</p> <p>When comparing two partially-filled arrays for equality, both arrays should be declared const.</p> <p>When deleting an element from a partially-filled array, it is an error if the index of the element to be removed is >= size.</p> <p>In a partially-filled array, the size represents the effective size of the array.</p> <p>When inserting elements into a partially-filled array, the array should not be declared const.</p>	
<p>[1701] Where are the characters "Hello" stored in memory?</p> <pre>char s1[1024] = "Hello"; void f() { const char *s2 = "Goodbye"; char s3[] = "CS 150"; }</pre> <p>stack heap static storage area (read-only) static-storage area (read/write)</p>	<p>static-storage area (read/write)</p>
<p>[1702] Where are the characters "Goodbye" stored in memory?</p> <pre>char s1[1024] = "Hello"; void f() { const char *s2 = "Goodbye"; char s3[] = "CS 150"; }</pre> <p>stack heap static storage area (read-only) static-storage area (read/write)</p>	<p>static storage area (read-only)</p>
<p>[1703] Where are the characters "CS 150" stored in memory?</p> <pre>char s1[1024] = "Hello"; void f() { const char *s2 = "Goodbye"; char s3[] = "CS 150"; }</pre> <p>stack heap static storage area (read-only) static-storage area (read/write)</p>	<p>stack</p>

<div><pre>char s1[1024] = "Hello"; void f() { const char *s2 = "Goodbye"; char s3[] = "CS 150"; }</pre></div> <div>stack heap static storage area (read-only) static-storage area (read/write)</div>	
<div>[1705] What happens here</div> <div><pre>void f() { char * s = "CS 150"; s[0] = 'X'; cout << s << endl; }</pre></div> <div>Prints "XS 150" Most likely crashes when run Code compiles without warnings Code fails to compile because "CS 150" is const</div>	<div>Most likely crashes when run</div>
<div>[1706] To process array-style (C) strings in C++, use the header:</div> <div><string> <cstring> <c-string> "cstring.h"</div>	<div><cstring></div>
<div>[1707] What happens here?</div> <div><pre>char * s = "CS150"; strcpy(s, "CS50"); cout << s << endl;</pre></div> <div>The code will not compile Code will compile (with warnings), but crash when run</div> <div>"CS50" "CS500" "CS150CS50"</div>	<div>Code will compile (with warnings), but crash when run.</div>
<div>[1708] What happens here?</div> <div><pre>char s[] = "CS150"; strcpy(s, "CS50"); cout << s << endl;</pre></div> <div>Crashes when run Undefined behavior</div> <div>"CS50" "CS500" "CS150CS50"</div>	<div>"CS50"</div>
<div>[1709] What happens here?</div> <div><pre>char s[] = "CS150"; strcat(s, "CS50"); cout << s << endl;</pre></div> <div>Crashes when run Undefined behavior</div> <div>"CS50" "CS500" "CS150CS50"</div>	<div>Undefined behavior</div>

Total combine1		Study	...
<pre>char s[50] = "CS150"; strcat(s, "CS50"); cout << s << endl;</pre> <p>Crashes when run Undefined behavior</p> <p>"CS50" "CS500" "CS150CS50"</p>			
<p>[1711] What happens here?</p> <pre>char s1[] = "CS150"; char *s2 = s1; s2[0] = 'X'; cout << s1 << endl;</pre> <p>"XS150" "CS150"</p> <p>Crashes when run Does not compile Undefined behavior</p>		"XS150"	
<p>[1712] What happens here?</p> <pre>char *s1 = "CS150"; char s2[] = s1; // C++ forbids converting a string constant to 'char*' s2[0] = 'X'; cout << s1 << endl;</pre> <p>"XS150" "CS150"</p> <p>Crashes when run Does not compile Undefined behavior</p>		Does not compile	
<p>[1713] What happens here?</p> <pre>char s1[] = "CS150", s2[10]; strcpy(s2, s1); s2[0] = 'X'; cout << s1 << endl;</pre> <p>"XS150" "CS150" Does not compile Crashes when run. Undefined behavior</p>		"CS150"	
<p>[1714] What happens here?</p> <pre>char s1[] = "CS150", s2[10]; strcpy(s1, s2); s2[0] = 'X'; cout << s1 << endl;</pre> <p>"XS150" "CS150" Does not compile Crashes when run. Undefined behavior</p>		Undefined behavior	
<p>[1715] What is true about a?</p> <pre>char a[] = "Sup?";</pre> <p>It is an array with sizeof 4 It is an array with sizeof 5 It is a C-string with strlen 5 It is a pointer to an array of 4 characters</p>		It is an array with sizeof 5	

Total combine1		Study	<div>1/1</div>
<div><pre>const char a = "dog", b = a; if (strcmp(a, b)) cout << "dog == dog" << endl; else cout << "dog != dog" << endl; dog != dog dog == dog Crashes when run Does not compile</pre></div>			
<div><div>[1717] What prints here?</div><pre>const char a = "dog", b = a; if (a == b) cout << "dog == dog" << endl; else cout << "dog != dog" << endl; dog != dog dog == dog Crashes when run Does not compile</pre></div>	<div>dog == dog</div>		
<div><div>[1718] What is the result of running this line of code?</div><pre>char s[] = "hi\0hey"; 3 chars 'h', 'i', '\0' stored in s. strlen(s) is 2. 6 chars, 'h','i','\0','h','e','y' stored in s. strlen(s) is 2. 7 chars, 'h','i','\0','h','e','y','\0' stored in s. strlen(s) is 2. 7 chars, 'h','i','\0','h','e','y','\0' stored in s. strlen(s) is 6. This is a syntax error.</pre></div>	<div>7 chars, 'h','i','\0','h','e','y','\0' stored in s. strlen(s) is 2.</div>		
<div><div>[1719] Which of these is a legal assignment?</div><pre>string name = "Houdini"; string str = c_str(name); char* cstr = name.c_str(); string* strp = name.c_str(); const char *cstr = c_str(name); const char *cstr = name.c_str();</pre></div>	<div>const char *cstr = name.c_str();</div>		
<div><div>[1720] Which line makes the comment correct?</div><pre>char s[50]; char *t = "ac"; // Make s into a C-string "ac" s = t; s = "ac"; s[0] = t[0]; s[1] = t[1]; s[2] = t[2]; None of these s[0] = t[0]; s[1] = t[1];</pre></div>	<div>s[0] = t[0]; s[1] = t[1]; s[2] = t[2];</div>		
<div><div>[1721] Which lines create the C-string "hello"?</div><pre>1. char s[10] = "hello"; 2. char s[10] = {'h','e','l','l','o'}; 3. char s[] = {'h','e','l','l','o','0'}; 4. char s[5] = "hello"; 5. char s[] = "hello"; 1, 2, 3, 5 1, 2, 5 All of them 1, 3 1, 5</pre></div>	<div>1, 2, 5</div>		
<div><div>[1722] Which lines contains exactly two characters?</div><pre>1. "\n" 2. '\n' 3. "n" 4. "/n" 5. 'n' 1, 3, 5 1, 2, 4 All of them 1, 3, 4 1, 3</pre></div>	<div>1, 3</div>		



<pre>void stringCopy(char *p, const char *q) { while ((*p = *q) != '\0') { p++; q++; } }</pre> <p>No, because there is no *p = '\0'; after the loop</p> <p>No, because the comparison should be against 0, not against '\0'</p> <p>No, because the condition accidentally used = instead of ==</p> <p>Yes, the terminator is copied as the condition fails</p> <p>No, because there is no actual copy of characters into p at all</p>	
<p>[1724] Which while condition makes this function correct?</p> <pre>int stringComp(const char *s1, const char * s2) { while (. .) { s1++; s2++; } return *s1 - *s2 }</pre> <p>*s1 != *s2</p> <p>*s1 == *s2</p> <p>*s1 && *s2</p> <p>*s1 == *s2 *s1 *s2</p> <p>*s1 == *s2 && *s1 && *s2</p>	<p>*s1 == *s2 && *s1 && *s2</p>
<p>[1725] Which library function performs an equivalent operation on C-strings?</p> <pre>string s1 = "Hello"; string s2 = "World"; s1 = s1 + s2;</pre> <p>strlen()</p> <p>strcpy()</p> <p>strcmp()</p> <p>strcat()</p> <p>None of these</p>	<p>strcat()</p>
<p>[1726] Which library function performs an equivalent operation on C-strings?</p> <pre>string s1 = "Hello"; string s2 = "World"; s1 = s2;</pre> <p>strlen()</p> <p>strcpy()</p> <p>strcmp()</p> <p>strcat()</p> <p>None of these</p>	<p>strcpy()</p>
<p>[1727] Which library function performs an equivalent operation on C-strings?</p> <pre>string s1 = f0, s2 = f0; if (s1 < s2) . . .</pre> <p>strlen()</p> <p>strcpy()</p> <p>strcmp()</p> <p>strcat()</p> <p>None of these</p>	<p>strcmp()</p>
<p>[1728] Which library function performs an equivalent operation on C-strings?</p> <pre>string s = mystery(); if (s.size() > 3) . . .</pre> <p>strlen()</p> <p>strcpy()</p> <p>strcmp()</p> <p>strcat()</p> <p>None of these</p>	<p>strlen()</p>



The characters for the C-string `char * s1 = "hello";` are stored in user memory and may be modified.

`strcmp(s1, s2)` returns true if s1 and s2 contain the same characters.

The `strlen()` function returns the allocated size of a C-string allocated as an array.

The C-string type is part of the standard library, not built into the C++ language.

C-string assignment uses the `=` operator.

The length of a C-string is stored explicitly in its `length` data member

The allocated size for the C-string `char s1[1024] = "hello";` is 6 characters, while the effective size is 5 characters.

C-string assignment uses the `strcat()` function.

The `strcat()` function cannot overflow the storage allocated for the destination buffer.

The `strncpy()` function always appends a trailing NUL when the copy is finished.

`strcmp(s1, s2)` returns a negative number if s1 is lexicographically "greater than" s2.

The `sizeof` operator returns the effective size of a C-string allocated as an array.

The `strncpy()` function is straightforward and easy to use.

`strcmp(s1, s2)` returns a positive number if s1 is lexicographically "less than" s2.

You can compare two C-strings, s1 and s2, by using the `==` operator.

C-strings use the `+` operator for concatenation.

C-strings are `char` pointers to the first character in a sequence of characters, terminated with a `'\0'` character.

When writing programs that interact with your operating system, either Windows, Mac OSX or Linux, you will normally use the C++ library `string` type, rather than the older C-string type.

The C-string literal `"cat"` contains 3 characters.

The `strcpy()` function expands the destination string to make sure it is large enough to hold the source string.



You can compare two C-strings, s1 and s2, by using the strcmp() function.

C-strings are character arrays that rely on a special embedded sentinel value, the character with the ASCII code 0.

The allocated size for the C-string char s1[] = "hello"; is 6 characters, while the effective size is 5 characters.

The sizeof operator returns the allocated size of a C-string allocated as an array.

The effective size of the C-string char * s1 = "hello"; is 5 characters, but 6 characters are used for storage.

strcmp(s1, s2) returns a positive number if s1 is lexicographically "greater than" s2.

C-strings use the strcat() function for concatenation.

The strlen() function returns the effective size of a C-string.

C-strings are often needed to interoperate with legacy C libraries.

When writing programs that interact with your operating system facilities, either Windows, Mac OSX or Linux, you will normally use C-strings instead of the C++ library string type.

The characters for the C-string char s1[] = "hello"; are stored in user memory and may be modified.

C-strings are char pointers to the first character in a sequence of characters, terminated with a '\0' character.

C-string functions may be more efficient than C++ string member functions.

strcmp(s1, s2) returns a negative number if s1 is lexicographically "less than" s2.

Given the C-string char * s3 = "hello"; strlen(s3) returns 5.

C-string assignment uses the strcpy() function.

strcmp(s1, s2) returns 0 if s1 and s2 contain the same characters.

The C-string type is built into the C++ language, not defined in the standard library.

The strcpy() function always appends a trailing NUL when the copy is finished.

The strncpy() function can be used to make sure that you don't copy more characters than necessary.

Programs written for embedded devices often use C-strings rather than the C++ library string type.

The length of a C-string is never stored explicitly

The C-string literal "cat" contains 4 characters.

The strncat() function allows you to limit the maximum number of characters that are concatenated.

The character with the ASCII code 0 is called the NUL character

[1801] Which of these is a 2D array?

int d[][]
int *b[2];
int a[][2];
int c[2][2];

int c[2][2];

[1802] Which function prototype could process a 2D array?

void f(int **a);
void f(int[][] a);
void v(int a[][]);
void f(int a[2][]);
void f(f(int a[][2]);

void f(f(int a[][2]);

[1803] What prints? Assume 4 bytes per int.

int a[2] = {0};
cout << sizeof(a) << endl;

16
12
4
8

Illegal declaration. Does not compile.

8

Total combine1		Study	
<pre>int a[][2] = {{0},{0}}; cout << sizeof(a) << endl;</pre> <div>4 12 16 8</div> <p>Illegal declaration. Does not compile.</p>			
<p>[1805] What prints? Assume 4 bytes per int.</p> <pre>int a[][2] = {1, 2, 3}; cout << sizeof(a) << endl;</pre> <div>8 12 4 16</div> <p>Illegal declaration. Does not compile.</p>	16		
<p>[1806] What prints? Assume 4 bytes per int.</p> <pre>int a[][] = {{1, 2}, {3, 4}}; cout << sizeof(a) << endl;</pre> <div>4 12 16 8</div> <p>Illegal declaration. Does not compile.</p>	Illegal declaration. Does not compile.		
<p>[1807] What prints?</p> <pre>int a[4][2] = {1, 2, 3, 4, 5, 6, 7}; cout << a[2][1] << endl;</pre> <div>Undefined (out of bounds) 6 Illegal declaration. Does not compile. 5 4 // 0 1 // 0 { 1, 2 // 1 3, 4 // 2 5, 6 // 3 7, 0 }</div>	6		
<p>[1808] Which one of the following statements is the correct definition for a two-dimensional array of 20 rows and 2 columns of the type integer?</p> <p>int num[2, 20] int num[2][2]; int num[20][2]; None of these int num[20, 2];</p>	int num[20][2];		
<p>[1809] Which statement displays the value 24 from the 2D array initialized here?</p> <pre>int a[2][3] = { { 13, 23, 33 }, { 14, 24, 34 } }; cout << a[2][2]; cout << a[1][2]; cout << a[1][1]; cout << a[2][1]; None of these</pre>	cout << a[1][1];		
<p>[1810] Which value of a is stored in the val variable?</p> <pre>auto val = a[0][2];</pre> <p>The value in the first row and the third column The value in the third row and the first column The value in the first row and the second column None of these The value in the first row and the first column</p>	The value in the first row and the third column		

Total combine1		Study	
<pre>cout << a[3][2]; cout << a[2][1]; None of these cout << a[2][3]; cout << a[1][2];</pre>			
<p>[1812] What prints when this runs?</p> <pre>int a[2][3] = {1, 2, 3, 4, 5, 6}; cout << a[0][2] + a[1][2] << endl;</pre> <p>5 10 7 8 9</p> <p>// 0, 1, 2 // 0 { 1, 2, 3 // 1 4, 5, 6 }</p>		9	
<p>[1813] What is the value of a[1][1] after this runs?</p> <pre>int cnt = 0, a[2][3]; for (int i = 0; i < 3; i++) for (int j = 0; j < 2; j++) a[j][i] = ++cnt;</pre> <p>6 4 2 3 5</p>		4	
<p>[1814] What is the value of a[1][2] after this runs?</p> <pre>int cnt = 0, a[2][3]; for (int i = 0; i < 3; i++) for (int j = 0; j < 2; j++) a[j][i] = ++cnt;</pre> <p>4 6 2 5 3</p>		6	
<p>[1815] What is the value of a[0][2] after this runs?</p> <pre>int cnt = 0, a[2][3]; for (int i = 0; i < 3; i++) for (int j = 0; j < 2; j++) a[j][i] = ++cnt;</pre> <p>6 4 2 5 3</p>		5	
<p>[1816] What prints?</p> <pre>int a[2][3] = {{3,2,3}}; cout << a[0][0] << a[1][0] << endl;</pre> <p>00 Code does not compile 31 30 33</p>		30	
<p>[1817] What prints?</p> <pre>int a[3][2] = {{3,2,3}}; // too many initializers for 'int [2]' cout << a[0][0] << a[1][0] << endl;</pre> <p>00 30 Code does not compile 33 31</p>		Code does not compile	



<pre>int a[3][2] = {3,2,3}; cout << a[0][0] << a[1][0] << endl;</pre> <p>Code does not compile</p> <p>31</p> <p>33</p> <p>00</p> <p>30</p>	
<p>[1819] What prints?</p> <pre>int cnt = 0, a[4][5]; for (int i = 0; i < 5; i++) for (int j = 0; j < 4; j++) a[j][i] = cnt++; cout << a[1][2] << endl;</pre> <p>11</p> <p>9</p> <p>19</p> <p>8</p> <p>14</p>	9
<p>[1820] What prints?</p> <pre>int cnt = 0, a[4][5]; for (int i = 0; i < 5; i++) for (int j = 0; j < 4; j++) a[j][i] = cnt++; cout << a[2][3] << endl;</pre> <p>19</p> <p>14</p> <p>11</p> <p>9</p> <p>8</p>	14
<p>[1821] What prints?</p> <pre>int cnt = 0, a[4][5]; for (int i = 0; i < 5; i++) for (int j = 0; j < 4; j++) a[j][i] = cnt++; cout << a[3][2] << endl;</pre> <p>14</p> <p>11</p> <p>9</p> <p>8</p> <p>19</p>	11
<p>[1822] What prints?</p> <pre>int cnt = 0, a[4][5]; for (int i = 0; i < 5; i++) for (int j = 0; j < 4; j++) a[j][i] = cnt++; cout << a[3][2] << endl;</pre> <p>11</p> <p>8</p> <p>9</p> <p>14</p>	11
<p>[1823] How many rows are in this array?</p> <pre>int a[2][3];</pre> <p>6</p> <p>3</p> <p>5</p> <p>4</p> <p>2</p>	2
<p>[1824] How many columns are in this array?</p> <pre>int a[2][3];</pre> <p>3</p> <p>6</p> <p>4</p> <p>5</p> <p>2</p>	3



<pre>int a[2][3]; 3 2 5 4 6</pre>	
<p>[1826] How many (int[]) elements are in this array?</p> <pre>int a[2][3]; 5 3 2 4 6</pre>	2
<p>[1827] What is the correct version of main() if you wish to process command-line arguments?</p> <pre>int main(int argc, char* argv[]) int main(int argc[], char * argv) int main(char *argc, int argv[]) int main(char argv[], int argc) int main(string args[])</pre>	<pre>int main(int argc, char* argv[])</pre>
<p>[1828] What is true about the command line in C++?</p> <p>I. The first argument is the name of the program II. Command line arguments are passed in an array III. Use main(int argc, char* argv[])</p> <p>I and III All of these are true I only II and III II only</p>	All of these are true
<p>[1829] Why is the command-line argc always at least 1?</p> <p>Because argv[0] is the name of the program running Because the argv[] array is a special case that starts at 1 Because argv[0] is unused Because argv[0] is a pointer named this It is not. If there are no arguments passed, then argc is 0</p>	Because argv[0] is the name of the program running
<p>[1830] The program a.out is run like this:</p> <pre>./a.out alex brent chris rodger 32 33 44 78</pre> <p>argc is 9 and argv[0] is "/a.out" argc is 8 and argv[0] is "alex" argc is 9 and argv[0] is "alex" argc is 9 and argv[0] is "/a" argc is 8 and argv[0] is "/a.out"</p>	argc is 9 and argv[0] is "/a.out"
<p>[1831] What prints?</p> <pre>int a[5][3] = { { 1, 2, 3}, { 4, 5, 6}, { 7, 8, 9}, {10, 11, 12}, {13, 14, 15} }; int *p = &a[0][0]; cout << p[1][2] << endl; // invalid types 'int[int]' for array subscript Undefined (out of bounds) 6 2 Illegal; will not compile An address</pre>	Illegal; will not compile



<pre>int a[5][3] = { { 1, 2, 3}, { 4, 5, 6}, { 7, 8, 9}, {10, 11, 12}, {13, 14, 15} }; int *p = &a[0][0]; cout << *p << endl;</pre> <p>4 Illegal; will not compile 1 An address Undefined (out of bounds)</p>	
<p>[1833] What prints?</p> <pre>int a[5][3] = { { 1, 2, 3}, { 4, 5, 6}, { 7, 8, 9}, {10, 11, 12}, {13, 14, 15} }; int *p = &a[0][0]; cout << (p + 5) << endl;</pre> <p>4 An address Illegal; will not compile 1 Undefined (out of bounds)</p>	An address
<p>[1834] What prints?</p> <pre>int a[5][3] = { { 1, 2, 3}, { 4, 5, 6}, { 7, 8, 9}, {10, 11, 12}, {13, 14, 15} }; int *p = &a[0][0]; cout << p[10] << endl;</pre> <p>Illegal; will not compile An address 10 11 Undefined (out of bounds)</p>	11
<p>[1835] What prints?</p> <pre>int a[5][3] = { { 1, 2, 3}, { 4, 5, 6}, { 7, 8, 9}, {10, 11, 12}, {13, 14, 15} }; int *p = &a[0][0]; cout << (p + 5 * 2)[1] << endl;</pre> <p>13 12 11 Undefined (out of bounds) Illegal; will not compile</p>	12

Total combine1		Study	<div>1/1</div>
<pre>int x = 0; int a[2][3] = {{1, 2, 3}, {4, 5, 6}}; for (auto r : a) for (auto c : r) x++; // 'r' was not declared in this scope cout << x << endl;</pre>			
Undefined (out of bounds) 6 Illegal; will not compile 2 3			
[1837] What prints? <pre>int x = 0; int a[2][3] = {{1, 2, 3}, {4, 5, 6}}; for (const auto& r : a) for (const auto& c : r) x++; cout << x << endl;</pre> 3 2 Undefined (out of bounds) 6 Illegal; will not compile	6		
[1838] What prints? <pre>int x = 0; int a[2][3] = {{1, 2, 3}, {4, 5, 6}}; for (const auto& r : a) for (const auto& c : r) x += c; cout << x << endl;</pre> 21 Undefined (out of bounds) 15 6 Illegal; will not compile	21		
[1839] What prints? <pre>int x = 0; int a[2][3] = {{1, 2, 3}, {4, 5, 6}}; for (const auto& r : a) for (const auto& c : r) x = c; cout << x << endl;</pre> 21 Undefined (out of bounds) 6 15 Illegal; will not compile	6		
You can pass the first row of the 2D array int a[3][3] to the function f(int *a, size_t n) by calling f(a[0], 3).		True	
You can pass the 2D array int a[3][3] to the function f(int *a, size_t r, size_t c) by calling f(&a[0][0], 3, 3).		True	
Physically, a 2D array is stored as a single linear, contiguous array with the elements for each column following the elements for the previous column in memory.		False	
Command line arguments that start with a hyphen are usually called switches.		True	
You can use a range-based loop on a 2D array.		True	
You can pass the 2D array int a[3][3] to the function f(int a[3][], size_t n) by calling f(a, 3).		False	
Physically, a 2D array is stored as a single linear, contiguous array with the elements for each row following the elements for the previous row in memory.		True	
A 2D array address expression is the equivalent of: (address + (row height * col))		False	
You cannot use a range-based loop on a 2D array.		False	
You can pass the first column of the 2D array int a[3][3] to the function f(int *a, size_t n) by calling f(a[0], 3).		False	
You can pass the 2D array int a[3][3] to the function f(int a[][3], size_t n) by calling f(a, 3).		True	
In a 2D array the first subscript represents the rows and the second the columns.		True	



A 2D array address expression is the equivalent of: (address + (row width + col))	True
When initializing a 2D, each row must have its own set of braces.	False
When passing a 2D array to a function, the array parameter must explicitly list the size for all dimensions except for the last, like: void f(int a[3][], size_t n);	False
A 2D array is a 1D array whose elements are also 1D arrays.	True
The rules for partial initialization of a 2D array can be changed by adding braces around interior array elements.	True
You can pass the 2D array int a[3][3] to the function f(int a[][], size_t r, size_t c) by calling f(a, 3, 3).	False
Your operating system's command processor is known as the shell.	True
When initializing a 2D, each column must have its own set of braces.	False
You can pass the 2D array int a[3][3] to the function f(int *a, size_t r, size_t c) by calling f(a, 3, 3).	False
Conceptually, a 2D array is rectangular grid of columns and rows.	True
Physically, a 2D array is stored as a rectangular grid of columns and rows.	False
When passing a 2D array to a function, the array parameter must explicitly list the size for all dimensions except for the first, like: void f(int a[][3], size_t n);	True
In a 2D array the first subscript represents the columns and the second the rows.	False
On the command line, argc is the count of arguments including the program itself.	True
<div>[1901] The variable p is located:</div> <div>void f() { int *p = new int; }</div> <div>in the static storage area None of these on the heap on the stack</div>	on the stack
<div>[1902] The variable *p is located:</div> <div>void f() { int *p = new int; }</div> <div>on the heap None of these in the static storage area on the stack</div>	on the heap
<div>[1903] The variable *p:</div> <div>void f() { int *p = new int; }</div> <div>Stores a memory address Stores the value 0 It's uninitialized</div>	It's uninitialized
<div>[1904] The variable p:</div> <div>void f() { int *p = new int; }</div> <div>stores the value 0 stores a memory address None of these is uninitialized</div>	stores a memory address



<div><div><div>void f() { int *p = new int{42}; }</div><div></div></div><div><div></div><div></div></div><div><div>It's undefined → Code does not compile</div><div></div></div><div><div>Stores a memory address</div><div></div></div><div><div>Stores the value 42 in all versions of C++</div><div></div></div><div><div>Stores the value 42 in C++11 only</div><div></div></div><div><div>It's uninitialized</div><div></div></div></div>	
<div><div><div>[1906] The variable *p: void f() { int *p = new int(42); }</div><div></div></div><div><div></div><div></div></div><div><div>It's undefined → Code does not compile. Stores the value 42 in all versions of C++ Stores a memory address It's uninitialized Stores the value 42 in C++11 only</div><div></div></div></div>	<div>Stores the value 42 in all versions of C++</div>
<div><div><div>[1907] The variable *p: void f() { int *p = new int{}; }</div><div></div></div><div><div></div><div></div></div><div><div>Stores a memory address It's uninitialized Stores the value 0 in all versions of C++ Stores the value 0 in C++11 only It's undefined → Code does not compile.</div><div></div></div></div>	<div>Stores the value 0 in C++11 only</div>
<div><div><div>[1908] The variable *p: void f() { int *p = new int = {42}; }</div><div></div></div><div><div></div><div></div></div><div><div>Stores the value 42 in all versions of C++ It's undefined → Code does not compile. It's uninitialized Stores a memory address Stores the value 42 in C++11 only</div><div></div></div></div>	<div>It's undefined → Code does not compile.</div>
<div><div><div>[1909] The variable *p: void f() { string *p = new string; }</div><div></div></div><div><div></div><div></div></div><div><div>It's undefined → Code does not compile Stores an empty string Stores nullptr It's uninitialized Stores a memory address</div><div></div></div></div>	<div>Stores an empty string</div>


Total combine1		Study	
<div><div><div>void f() { int *p = new int[42]; }</div><div></div></div><div><div>It's undefined Code → does not compile The first element of an array of 42 uninitialized ints A single int with the value 42 The first element of an array of 42 ints with the value 0</div><div></div></div></div>			
<div><div><div>[1911] The variable p points to: void f() { int *p = new int[42](); }</div><div></div></div><div><div>The first element of an array of 42 uninitialized ints The first element of an array of 42 ints with the value 0 A single int with the value 42 It's undefined → Code does not compile</div><div></div></div></div>	The first element of an array of 42 ints with the value 0		
<div><div><div>[1912] The variable p points to: void f() { int *p = new int[3]{1, 2, 3}; }</div><div></div></div><div><div>is undefined. Code does not compile. the first element of an array of 3 uninitialized ints a single int with the value 1 the first element of an array of 3 ints with the values 1,2,3</div><div></div></div></div>	the first element of an array of 3 ints with the values 1,2,3		
<div><div><div>[1913] The variable p points to: void f() { int *p = new int[3] = {1, 2, 3}; }</div><div></div></div><div><div>the first element of an array of 3 ints with the values 1,2,3 the first element of an array of 3 uninitialized ints a single int with the value 1 is undefined. Code does not compile</div><div></div></div></div>	is undefined. Code does not compile.		
<div><div><div>[1914] Examine this code. What goes on the blank line? void f() { int *p = new int[3]{1, 2, 3}; ... } delete *p; delete[] p; delete p[3]; None of these is correct delete p;</div><div></div></div></div>	delete[] p;		
<div><div><div>[1915] Examine this code. What goes on the blank line? void f() { int *p = new int[3]{1, 2, 3}; ... } delete p[]; delete p; delete p[3]; None of these is correct delete[] *p;</div><div></div></div></div>	None of these is correct		

Total combine1		Study	
<div><div><div>void f() { int *p = new int[3]{rand(), rand(), rand()}; if (p[1] != 0 && p[2] != 0) cout << p[0] / p[1] / p[2] << endl; delete[] p; }</div></div></div>			
<div><div><div>[1922] This code: void f() { int *p = new int[3]{rand(), rand(), rand()}; if (p[1] != 0 && p[2] != 0) delete[] p; cout << p[0] / p[1] / p[2] << endl; }</div></div></div> <div><div>has a dangling pointer</div></div>			
<div><div><div>[1923] This code: int * f() { int a[] = {1, 2, 3}; return &a[1]; }</div></div></div> <div><div>None of these has a dangling pointer has a syntax error has a memory leak has a double delete</div></div>			
<div><div><div>[1924] This code: void f() { int *p = new int[3]{rand(), rand(), rand()}; if (p[1] != 0 && p[2] != 0) delete[] p; else cout << p[0] / p[1] / p[2] << endl; delete[] p; }</div></div></div> <div><div>has a double delete None of these has a dangling pointer has a syntax error has a memory leak</div></div>			
<div><div><div>[1925] To use any of C++ smart pointer types, include the header: <memory> <ptr> <new> <smart_ptr> <alloc></div></div></div>		<memory>	
<div><div><div>[1926] Which line correctly creates a smart pointer that points to the variable x? int x = 42; unique_ptr<int>(&x); make_shared<int>(x); unique_ptr<int>(&x); None of these shared_ptr<int>(&x);</div></div></div>		None of these	



<pre>int x = 42; unique_ptr<int[]>(&x); None of these shared_ptr<int>(&x); unique_ptr<int>(&x); make_shared<int>(x);</pre>	
<p>[1928] What does this code print?</p> <pre>int main() { auto p1 = make_shared<int>(42); auto p2 = p1; cout << *p1 << endl; cout << *p2 << endl; (*p2)++; cout << *p1 << endl; }</pre> <p>424343 Does not compile (illegal) 424242 Undefined behavior 424243</p>	<p>424243</p>
<p>[1929] Given this declaration, which line below is illegal?</p> <pre>auto p1 = make_shared<int>(42); cout << *p1 << endl; (*p1)++; delete p1; None of these are illegal auto p2 = p1;</pre>	<p>delete p1;</p>
<p>[1930] Given this declaration, which line below is illegal?</p> <pre>auto p1 = unique_ptr<int>(new int{42}); (*p1)++; p1.release(); None of these are illegal auto p2 = p1; cout << *p1 << endl;</pre>	<p>auto p2 = p1;</p>
<p>[1931] What does this code print?</p> <pre>int main() { auto p1 =unique_ptr<int>(new int{42}); cout << *p1; auto p2 = p1.release(); cout << *p2; (*p2)++; cout << *p2; }</pre> <p>Undefined behavior 424343 424243 424242 Does not compile (illegal)</p>	<p>424243</p>
<p>[1932] What does this code print?</p> <pre>int main() { auto p1 =unique_ptr<int>(new int{42}); cout << *p1; auto p2 = p1; cout << *p2; (*p2)++; cout << *p2; }</pre> <p>Does not compile (illegal) 424242 424243 Undefined behavior 424343</p>	<p>Does not compile (illegal)</p>

Total combine1		Study	<div>100%</div>
<pre>int main() { auto p1 =unique_ptr<int>(new int{42}); cout << *p1; auto p2 = p1.release(); 🌸 cout << *p2; (*p2)++; cout << *p1; 🌸 Isn't called correctly }</pre> <p>Does not compile (illegal) Undefined behavior 424242 424243 424343</p>			
<p>[1934] The member function get() returns the raw pointer that a smart pointer contains. What does this code print?</p> <pre>int main() { auto p1 =unique_ptr<int>(new int{42}); cout << *p1; auto p2 = p1.release(); // Resets to NULL Pointer cout << *p2; (*p2)++; cout << p1.get() << endl; // Returns 0 }</pre> <p>424343 42430 Does not compile (illegal) Undefined behavior 42420</p>		42420	
A unique_ptr uses a reference count to manage how many pointers point to an object.		False	
To allocate memory on the heap, C++ uses the new operator.		True	
Memory for global variables is allocated when the program is loaded from disk. This is known as dynamic allocation.		False	
The statement new int{3}; allocates an array of three integers on the heap.		False	
The statement new int{3}; allocates a single initialized integer on the heap.		True	
Memory for local variables is allocated on the stack when their definitions are encountered during runtime. This is known as dynamic allocation.		False	
Requesting a block of memory from the operating system as the program runs is known as static allocation.		False	
Memory for global variables is allocated when the program is loaded from disk. This is known as static allocation.		True	
Smart pointers may point to objects allocated on the stack.		False	
Assuming p is a pointer to a single variable allocated on the heap, the statement delete[] p; returns the allocated memory back to the operating system for reuse.		False	
A pointer that goes out of scope before deleting the memory it points to is called a double delete.		False	
A pointer-like object that can be used to automatically manage memory allocated on the heap is called a smart pointer.		True	
Memory for global variables is allocated when the program is loaded from disk. This is known as automatic allocation.		False	
The release() function returns the raw pointer that a unique_ptr contains before seting the pointer to nullptr.		True	
Assuming p is a pointer to a single variable allocated on the heap, the statement delete p; sets the pointer to nullptr so that the memory can be reused for another allocation.		False	
Memory for local variables is allocated on the stack when their definitions are encountered during runtime. This is known as static allocation.		False	
The statement new int{3}; allocates an array of three integers on the heap		False	
Using a pointer to access the memory it points to after the pointer has been deleted is called a double delete.		False	

Total combine1		Study	
A pointer that goes out of scope before deleting the memory it points to is called a memory leak.		True	
If the new operator cannot allocate memory, C++ throws an exception.		True	
The statement new int{}; is a syntax error.		False	
Using a pointer to access the memory it points to after the pointer has been deleted is called a memory leak.		False	
The reset() function returns the raw pointer that a unique_ptr contains, before setting that pointer to nullptr.		False	
To transfer a unique_ptr to a vector, use push_back along with the move() function.		True	
The statement new int[3]{1, 2, 3}; allocates an array of three initialized integers on the heap.		True	
Requesting a block of memory from the operating system as the program runs is known as dynamic allocation.		True	
Assuming p is a pointer to the first variable in an array allocated on the heap, the statement delete[] p; returns the allocated memory back to the operating system for reuse.		True	
The statement new int[3] = {1, 2, 3}; is a syntax error.		True	
The statement new int[3](); allocates an array of three default-initialized integers on the heap.		True	
The statement new int[3]; allocates an array of three uninitialized integers on the heap.		True	
The statement new int{}; allocates a default-initialized integer on the heap.		True	
Freeing unused memory that was allocated elsewhere in your program is done in C++ using a garbage collector.		False	
The statement new int[3](); is a syntax error.		False	
Using a pointer to access the memory it points to after the pointer has been deleted is called a dangling pointer.		True	
A unique_ptr can refer to a dynamic array.		True	
A unique_ptr may transfer its ownership to another unique_ptr.		True	
Assuming p is a pointer to a single variable allocated on the stack, the statement delete p; returns the allocated memory back to the operating system for reuse.		False	
Freeing unused memory that was allocated elsewhere in your program is done in C++ using manual memory management.		True	
The release() function deletes the raw pointer that a unique_ptr contains, and then sets that pointer to a new value.		False	
Assuming p is a pointer to the first variable in an array allocated on the heap, the statement delete p; returns the allocated memory back to the operating system for reuse.		False	
A shared_ptr uses a reference count to manage how many pointers point to an object.		True	
A pointer that goes out of scope before deleting the memory it points to is called a dangling pointer.		False	
To allocate memory on the stack, C++ uses the new operator.		False	
The statement new int; allocates an uninitialized integer on the heap.		True	
Smart pointers automatically delete the memory they point to at the appropriate time.		True	
Assuming p is a pointer to a single variable allocated on the heap, the statement delete p; returns the allocated memory back to the operating system for reuse.		True	
If the new operator cannot allocate memory, C++ returns nullptr.		False	
The statement new int; allocates an uninitialized integer on the stack.		False	
The statement new int[3] = {1, 2, 3}; allocates an array of three initialized integers on the heap.		False	



A pointer-like object that can be used to automatically manage memory allocated on the heap is called a raw pointer.	False
Requesting a block of memory from the operating system as the program runs is known as automatic allocation.	False
<div><div>[2001] Which item is a mutator?</div><div><pre>class Alligator { public: Alligator(double w); void eat(); string toString() const; private: double weight; };</pre></div><div>None of these toString() weight eat() Alligator()</div></div>	eat()
<div><div>[2002] Which of these is a default constructor?</div><div><pre>class Alligator { public: Alligator(double w); void eat(); string toString() const; private: double weight; };</pre></div><div>None of these Alligator() toString() weight eat()</div></div>	None of these
<div><div>[2003] Which of these is a constructor?</div><div><pre>class Alligator { public: Alligator(double w); void eat(); string toString() const; private: double weight; };</pre></div><div>weight toString() Alligator() None of these eat()</div></div>	Alligator()
<div><div>[2004] Which of these is an accessor?</div><div><pre>class Alligator { public: Alligator(double w); void eat(); string toString() const; private: double weight; };</pre></div><div>toString() weight Alligator() None of these eat()</div></div>	toString()



<div><pre>class Alligator { public: Alligator(double w); void eat(); string toString() const; private: double weight; }; toString() Alligator() eat() weight All of these</pre></div>	
<div><p>[2006] What type of member function is eat()?</p><pre>class Alligator { public: Alligator(double w); void eat(); string toString() const; private: double weight; }; mutator None of these destructor accessor constructor</pre></div>	<div>mutator</div>
<div><p>[2007] What type of member function is toString()?</p><pre>class Alligator { public: Alligator(double w); void eat(); string toString() const; private: double weight; }; constructor None of these mutator accessor destructor</pre></div>	<div>accessor</div>
<div><p>[2008] What is true about user-defined types implemented using structures with public data members?</p><p>you cannot enforce the invariant properties of your types modifications to the character of the data members requires clients to rewrite code they may be more error-prone than types developed using classes clients can directly modify the data members of a variable</p></div>	<div>ANSWER → All of these</div>
<div><p>[2009] What is true about user-defined types implemented using classes with private data members?</p><p>clients can directly modify the data members of a variable it is not possible to create immutable objects All of these you can enforce the invariant properties of your types modifications to the character of the data members requires clients to rewrite code</p></div>	<div>you can enforce the invariant properties of your types</div>
<div><p>[2010] What is true about user-defined types implemented using classes with private data members?</p><p>clients can directly modify the data members of a variable you cannot enforce the invariant properties of your types it is not possible to create immutable objects All of these modifications to the character of the data members does not require clients to rewrite code</p></div>	<div>modifications to the character of the data members does not require clients to rewrite code</div>

<div>it is possible to create immutable objects</div> <div>All of these</div> <div>modifications to the character of the data members requires clients to rewrite code</div> <div>clients can directly modify the data members of a variable</div> <div>you cannot enforce the invariant properties of your types</div>	
<div>[2012] The _____ of a class specifies how clients interact with a class.</div> <div>public interface</div> <div>private implementation</div> <div>private interface</div> <div>public implementation</div> <div>None of these</div>	<div>public interface</div>
<div>[2013] What is f()?</div> <div>class X</div> <div>{</div> <div>public:</div> <div>X(int);</div> <div>void f() const;</div> <div>int g() const;</div> <div>void h(int);</div> <div>};</div> <div>None of these</div> <div>mutator</div> <div>destructor</div> <div>constructor</div> <div>accessor</div>	<div>None of these</div>
<div>[2014] What is g()?</div> <div>class X</div> <div>{</div> <div>public:</div> <div>X(int);</div> <div>void f() const;</div> <div>int g() const;</div> <div>void h(int);</div> <div>};</div> <div>None of these</div> <div>mutator</div> <div>accessor</div> <div>destructor</div> <div>constructor</div>	<div>accessor</div>
<div>[2015] What is h()?</div> <div>class X</div> <div>{</div> <div>public:</div> <div>X(int);</div> <div>void f() const;</div> <div>int g() const;</div> <div>void h(int);</div> <div>};</div> <div>mutator</div> <div>destructor</div> <div>constructor</div> <div>accessor</div> <div>None of these</div>	<div>mutator</div>
<div>[2016] What is X()?</div> <div>class X</div> <div>{</div> <div>public:</div> <div>X(int);</div> <div>void f() const;</div> <div>int g() const;</div> <div>void h(int);</div> <div>};</div> <div>accessor</div> <div>destructor</div> <div>mutator</div> <div>constructor</div> <div>None of these</div>	<div>constructor</div>



<div><div><div>class Val</div><div>{</div><div>int data_;</div><div>public:</div><div>Val(int);</div><div>int get() const;</div><div>void print() const;</div><div>};</div></div><div><div>void Val::get() { return data_; }</div><div>Val::Val(int n) { data_ = n; }</div><div>void Val::print() const { cout << data_; }</div><div>None of these</div><div>Val()</div><div>print()</div><div>data_</div><div>get()</div></div></div>	
<div><div><div>[2018] Which element is private?</div></div><div><div>struct Val</div><div>{</div><div>int data_;</div><div>public:</div><div>Val(int);</div><div>int get() const;</div><div>void print() const;</div><div>};</div></div><div><div>void Val::get() { return data_; }</div><div>Val::Val(int n) { data_ = n; }</div><div>void Val::print() const { cout << data_; }</div></div><div><div>Val()</div><div>None of these</div><div>get()</div><div>print()</div><div>data_</div></div></div>	<div><div>None of these</div></div>
<div><div><div>[2019] What is true about a mutator member function?</div></div><div><div>None of these</div><div>It changes one or more data members</div><div>It return information about an object's internal state</div><div>Its prototype ends with const</div><div>Its presence means that a class is immutable</div></div></div>	<div><div>It changes one or more data members</div></div>
<div><div><div>[2020] What is true about an accessor member function?</div></div><div><div>It is never used when a class is immutable</div><div>It returns information about an object's state</div><div>It changes one or more data members</div><div>Its prototype may not include the keyword const</div><div>None of these</div></div></div>	<div><div>It returns information about an object's state</div></div>
<div><div><div>[2021] Which of these are part of the implementation?</div></div><div><div>class Time {</div><div>public:</div><div>Time();</div><div>long get() const;</div><div>void set(long);</div><div>private:</div><div>long seconds;</div><div>};</div></div><div><div>The accessor and the mutator</div><div>The constructor</div><div>The data member seconds</div><div>All of these are part of the implementation</div><div>None of these are part of the implementation</div></div></div>	<div><div>The data member seconds</div></div>



<div><pre>class Time { Time(); long get() const; void set(long); private: long seconds; };</pre></div> <div><p>The accessor and the mutator</p><p>All of these are part of the implementation</p><p>None of these are part of the implementation</p><p>The data member seconds</p><p>The constructor</p></div>	
<div><p>[2023] What is the semantic error in this class definition?</p><pre>class Time { long seconds; public: Time(); long get() const; void set(long); };</pre></div> <div><p>get() should not have const at the end</p><p>seconds should be in the private section</p><p>get() is missing an argument</p><p>There is no semantic error.</p><p>set() is missing const at the end</p></div>	<p>There is no semantic error.</p>
<div><p>[2024] What is the semantic error in this class definition?</p><pre>class Time { long seconds; public: Time(); long get(); void set(long); };</pre></div> <div><p>There is no semantic error.</p><p>seconds should be in the private section</p><p>get() is missing an argument</p><p>set() is missing const at the end</p><p>get() is missing const at the end</p></div>	<p>get() is missing const at the end</p>
<div><p>[2025] What is the semantic error in this class definition?</p><pre>class Time { long seconds; public: Time(); long get() const; void set(long) const; };</pre></div> <div><p>seconds should be in the private section</p><p>get() is missing an argument</p><p>set() should not have const at the end</p><p>There is no semantic error.</p><p>get() should not have const at the end</p></div>	<p>set() should not have const at the end</p>
<div><p>[2026] What prints here?</p><pre>class Car { double speed; public: Car(); Car(double s); double get() const; }; Car::Car() { speed = 10; } Car::Car(double s) { speed = s; } double Car::get() const { return speed; } int main() { Car c1, c2(5); cout << c1.get() << c2.get() << endl; }</pre></div> <div><p>Does not compile; c1 is not an object</p><p>15</p><p>Undefined; c1 not initialized</p><p>05</p><p>105</p></div>	<p>105</p>



<pre>class Car { double speed; public: Car(); Car(double s); double get() const; }; Car::Car() { speed = 10; } Car::Car(double s) { speed = s; } double Car::get() const { return speed; } int main() { Car c1(), c2(5); cout << c1.get() << c2.get() << endl; }</pre> <p>Undefined; c1 not initialized 05 15 105 Does not compile; c1 is not an object</p>	
<p>[2028] A user-defined type created as a struct</p> <p>encapsulates its data to prevent accidental modification can be easily modified without affecting code that uses it. has its implementation as its interface is an interface paired with an implementation enforces type invariants</p>	<p>has its implementation as its interface</p>
<p>[2029] A Fraction denominator must not ever become 0. You can enforce this invariant through:</p> <pre>class Fraction { ... public: Fraction(int, int); Fraction get() const; Fraction set(int, int); };</pre> <p>the implementation of the accessor member the selection of data members the implementation of the mutator member by using the access modifier private in place of public the implementation of a destructor</p>	<p>the implementation of the mutator member</p>
<p>[2030] On the second line of this code, the object named myRadio is:</p> <pre>Radio myRadio(98.6, 8); cout << myRadio.frequency() << endl;</pre> <p>an implicit parameter an instance variable a function modifier an explicit parameter the function return value</p>	<p>an implicit parameter</p>
<p>[2031] The attributes of this class are model and price. In C++ terminology, these are called:</p> <pre>class Mobile { std::string model; double price; public: ... };</pre> <p>data members instance variables class variables data attributes fields</p>	<p>data members</p>



<div><pre>class Radio { public: Radio(); explicit Radio(double); Radio(double, int); double frequency() const; double frequency(double); };</pre></div> <div>constructor mutator data member accessor method</div>	
<div><p>[2033] In C++ terminology, frequency() is called a:</p><pre>class Radio { public: Radio(); explicit Radio(double); Radio(double, int); double frequency() const; double frequency(double); };</pre></div> <div>constructor method accessor data member mutator</div>	accessor
<div><p>[2034] In C++ terminology, the two members named frequency() are:</p><pre>class Radio { public: Radio(); explicit Radio(double); Radio(double, int); double frequency() const; double frequency(double); };</pre></div> <div>non-member functions methods constructors data members member functions</div>	member functions
<div><p>[2035] The default constructor is:</p><pre>class Radio { public: Radio(); explicit Radio(double); Radio(double, int); double frequency() const; double frequency(double); };</pre></div> <div>None of these Radio() Radio(double, int) Radio(double)</div>	Radio()
<div><p>[2036] There is no constructor for this class.</p><pre>class Integer { int value_ = 0; public: int get() const; int set(int n); };</pre></div> <div>You can create objects; value_ is initialized to 0 The code compiles, but you cannot create objects from this class You can create objects; value_ is uninitialized The code will not compile without a constructor</div>	You can create objects; value_ is initialized to 0



<pre>class Integer { int value_; public: int get() const; int set(int n); };</pre> <p>You can create objects; value_ is initialized to 0</p> <p>The code compiles, but you cannot create objects from this class</p> <p>You can create objects; value_ is uninitialized</p> <p>The code will not compile without a constructor</p>	
With classes, the public interface includes the member functions that allow clients to access object data in a safe way as well as the data members themselves.	False
In C++ you use the keyword public or private to create a section that indicates the access privileges of subsequent data members or member functions.	True
Member functions that change the state of an object are called constructors.	False
The member function <code>int hours() const;</code> provides read-write access to the hours property (however it is stored).	False
A class is an interface paired with an implementation.	True
Mutator member functions are allowed to read data members, but not change them.	False
Member functions that change the state of an object are called accessors.	False
Accessor member functions should always end in the keyword <code>const</code> .	True
If your class does not have a constructor, the compiler will synthesize a working constructor for you.	False
If a member function is in the private section of a class, it cannot be called from other member functions of the class.	False
The implementation of a class normally appears entirely inside the class <code>.cpp</code> file.	False
The implementation of a member function from the <code>Time</code> class would look something like this: <code>int Time::hours() const {...}</code> .	False
The interface of a class includes all items in the header file.	False
The interface of a class includes all public items in the header file.	True
When calling a member function, like <code>t.hours(3)</code> ; the address of the object <code>t</code> is passed to the function implicitly as the first parameter.	True
Member functions that initialize the data members of a new object are called mutators.	False
If you write a working constructor for your class, C++ will remove the synthesized default constructor.	True
The implementation of a member function from the <code>Time</code> class would look something like this: <code>int Time::hours() const {...}</code>	True
With classes, the public interface includes the member functions that allow clients to access object data in a safe way.	True
In C++ there is actually no difference between structures and classes.	False
A class definition ends with a semicolon.	True
The implementation of a class includes all private data members in the header file.	True
In C++ you use the keyword <code>public</code> or <code>private</code> before each data member or member function to indicate its access privileges.	False
In C++ there is actually no difference between structures and classes	False
Mutator member functions are allowed to read data members, but not change them	False
Programmers using class-derived objects, directly manipulate the data members of those objects.	False
Using structures for user-defined types means that you cannot change the data representation without affecting the users of your data type.	True
Using classes for user-defined types means that you cannot enforce restrictions on data member access.	False



Programmers using structure-derived variables, directly manipulate the data members of those variables.	True
You may add = default; to the prototype for a default constructor to retain the synthesized version in the presence of other overloaded constructors.	True
The implementation of a class normally appears partly inside the class .cpp file and partly inside the class .h file.	True
If your class does not have a constructor, the compiler will synthesize a default constructor for you.	True
If a member function is in the private section of a class, it can only be called by other member functions of the class.	True
The implementation of a member function from the Time class would look something like this: int hours() const {...}.	False
The two parts of a class are a private interface and a public implementation.	False
The public interface of a class consists of the prototypes of its member functions.	True
Mutator member functions are allowed to read data members and also change them.	True
A structure is an interface paired with an implementation.	False
Using structures for user-defined types means that you can enforce restrictions on data member access.	False
A class definition normally appears in a .h file.	True
In C++ the only difference between structures and classes is that member functions are public by default in structures.	True
Member functions that initialize the data members of a new object are called constructors.	True
Using structures for user-defined types means that you can change the data representation without affecting the users of your data type.	False
Accessor member functions are allowed to read data members, but not change them.	True
The two parts of a class are a public interface and a private implementation.	True
The member function int hours() const; provides read-only access to the hours property (however it is stored).	True
In C++ the only difference between structures and classes is that member functions are private by default in classes.	True
A constructor always has the same name as the class, and no return type.	True
The member function int& hours(); provides read-write access to the hours property (however it is stored).	True
Using structures for user-defined types means that you cannot enforce restrictions on data member access.	True
Using classes for user-defined types means that you can change the data representation without affecting the users of your class.	True
In C++ the only difference between structures and classes is that member functions are private by default in structures.	False
A constructor that takes no arguments is called the working constructor.	False



<p>A class definition normally appears in a .cpp file</p> <p>The member function <code>int& hours();</code> provides read-only access to the <code>hours</code> property (however it is stored)</p> <p>You may add <code>= default;</code> to the prototype of any constructor to allow the compiler to synthesize one for you</p> <p>The semicolon following a class definition is optional</p> <p>Member functions that initialize the data members of a new object are called accessors</p> <p>Using classes for user-defined types means that you cannot change the data representation without affecting the users of your class</p>	
<p>The keywords <code>public</code> and <code>private</code> are the C++ mechanism for defining interfaces and enforcing encapsulation</p> <p>Member functions that change the state of an object are called mutators</p> <p>A constructor that takes no arguments is called the default constructor</p> <p>Using classes for user-defined types means that you can enforce restrictions on data member access</p> <p>Accessor member functions are allowed to read data members and also change them</p>	<p>True</p>
<p>[2101] Which of these is not a property of an object (in the OOP sense)?</p> <p>Substitutability</p> <p>Identity</p> <p>State</p> <p>All of these are properties of an object</p> <p>Behavior</p>	<p>Substitutability</p>
<p>[2102] The _____ of an object consist of its attributes or characteristics, represented by the values stored in its data members.</p> <p>Identity</p> <p>State</p> <p>Class</p> <p>Behavior</p> <p>Object</p>	<p>State</p>
<p>[2103] A(n) _____ is a template or blueprint specifying the data attributes and behaviors for a group of similar objects.</p> <p>Behavior</p> <p>State</p> <p>Class</p> <p>Object</p> <p>Identity</p>	<p>Class</p>
<p>[2104] The _____ of an object is implemented by the object's member functions.</p> <p>Class</p> <p>Identity</p> <p>State</p> <p>Object</p> <p>Behavior</p>	<p>Behavior</p>
<p>[2105] Objects are _____ of a particular class.</p> <p>Instances</p> <p>Abstractions</p> <p>Identifiers</p> <p>Interfaces</p> <p>Encapsulations</p>	<p>Instances</p>
<p>[2106] _____ is the Object-Oriented design principle and technique that enforces data hiding.</p> <p>Abstraction</p> <p>Inheritance</p> <p>Dynamic Binding</p> <p>Polymorphism</p> <p>Encapsulation</p>	<p>Encapsulation</p>

Total combine1		Study	
attributes and behaviors.			
Abstraction Inheritance Dynamic Binding Polymorphism Encapsulation			
[2108] _____ is the Object-Oriented design feature that allows you to write programs in terms of an "ideal" class, but substitute or plug-in related objects that act in different ways when your program runs.		Polymorphism	
Inheritance Polymorphism Dynamic Binding Abstraction Encapsulation			
[2109] The working constructor for this class is: class Radio { public: Radio(); explicit Radio(double); Radio(double, int); double frequency() const; double frequency(double); }; Radio(double, int) Radio(double) Radio() None of these		Radio(double, int)	
[2110] The conversion constructor for this class is: class Radio { public: Radio(); explicit Radio(double); Radio(double, int); double frequency() const; double frequency(double); }; None of these Radio(double) Radio(double, int) Radio()		Radio(double)	
[2111] The copy constructor for this class is: class Radio { public: Radio(); explicit Radio(double); Radio(double, int); double frequency() const; double frequency(double); }; Radio(double, int) None of these Radio(double) Radio()		None of these	
[2112] What statement about constructors is false? All constructors are passed a pointer argument Constructors have no return type You must write at least one constructor for every class Constructors may take arguments Classes may have more than one constructor		You must write at least one constructor for every class	
[2113] Which members often use the modifier explicit in their declaration? The copy constructor The conversion constructor The default constructor None of these The working constructor		The conversion constructor	



<pre>#include <string> class Xynoid { double a; int b; std::string c; }; int main() { Xynoid x; }</pre> <p>Does not compile. Compiles and links. Two members uninitialized Compiles and links. All members initialized. Compiles but does not link. Compiles and links. All members uninitialized</p>	
<p>[2115] The following code:</p> <pre>#include <string> class Xynoid { double a{3.14}; int b = 42; std::string c; }; int main() { Xynoid x; }</pre> <p>Compiles and links. All members uninitialized Does not compile. Compiles and links. Two members uninitialized Compiles and links. All members initialized Compiles but does not link.</p>	Compiles and links. All members initialized
<p>[2116] The following code:</p> <pre>#include <string> class Xynoid { double a{3.14}; int b = 42; std::string c; public: Xynoid(double x, int y, std::string z); }; int main() { Xynoid x; }</pre> <p>Compiles and links. Two members uninitialized Compiles but does not link. Compiles and links. All members uninitialized Compiles and links. All members uninitialized Does not compile.</p>	Does not compile.
<p>[2117] The following code:</p> <pre>#include <string> class Xynoid { double a{3.14}; int b = 42; std::string c; public: Xynoid() = default; Xynoid(double x, int y, std::string z); }; int main() { Xynoid x; Xynoid z(1, 2, "fred"); }</pre> <p>Does not compile. Compiles and links. All members uninitialized Compiles but does not link. Compiles and links. All members initialized Compiles and links. Two members uninitialized</p>	Compiles but does not link



```
#include <string>
class Xynoid {
double a{3.14};
int b = 42;
std::string c;
public:
Xynoid() = default;
Xynoid(double x, int y, std::string z);
};
```

```
Xynoid::Xynoid(double x, int y, std::string z)
: c(z), b(y), a(x) { }
```

Constructor parameters are in the wrong order
Initializers use the wrong parameter values
There is no error. It is fine.
Initializers are in the wrong order.
There is no code in the body of the constructor

[2119] What happens here?

```
#include <iostream>
using namespace std;

class Dog {
int age_ = 7;
public:
Dog(int a);
int get() const;
};
Dog::Dog(int a): age_(a) { }
int Dog::get() const { return age_; }
```

```
int main()
{
Dog a(5);
Dog b(a);
Dog c = 10;
cout << a.get() << b.get() << c.get() << endl;
}
```

Line Dog b(a); does not compile. No suitable constructor.
Segmentation fault when line Dog c = 7 run.
Compiles, links: prints 5510
Compiles, links: prints 5710
Line Dog c = 10; do

Compiles, links: prints 5510

[2120] What happens here?

```
#include <iostream>
using namespace std;

class Dog {
int age_ = 7;
public:
explicit Dog(int a);
int get() const;
};
Dog::Dog(int a): age_(a) { }
int Dog::get() const { return age_; }
```

```
int main()
{
Dog a(5);
Dog b(a);
Dog c = 10;
cout << a.get() << b.get() << c.get() << endl;
}
```

Line Dog b(a); does not compile. No suitable constructor.
Segmentation fault when line Dog c = 7 run.
Line Dog c = 10; does not compile. Wrong type used for initializer.
Compiles, links: prints 5710
Compiles, links: prints 5510

Line Dog c = 10; does not compile. Wrong type used for initializer.



<pre>#include <string> #include <iostream> using namespace std; class Cat { string name_; public: Cat(const string& n); string get() const; }; Cat::Cat(const string& n): name_(n) {} string Cat::get() const { return name_; }</pre> <p>int main()</p> <p>{</p> <p>string s = "Bill";</p> <p>Cat b;</p> <p>b = s;</p> <p>cout << b.get() << endl;</p> <p>}</p> <p>Line beginning with: string Cat::get should not have const in implementation.</p> <p>Line Cat b; does not compile. No suitable constructor.</p> <p>Line beginning with: Cat::Cat should not have empty body</p> <p>Line b = s; does not compile. Type mismatch</p> <p>The does compile; it prints "Bill".</p>	
<p>[2122] What happens with this code?</p> <pre>#include <string> #include <iostream> using namespace std; class Cat { string name_; public: Cat(); Cat(const string& n); string get() const; }; Cat::Cat(const string& n): name_(n) {} string Cat::get() const { return name_; }</pre> <p>int main()</p> <p>{</p> <p>string s = "Bill";</p> <p>Cat b;</p> <p>b = s;</p> <p>cout << b.get() << endl;</p> <p>}</p> <p>The does compile; it prints "Bill".</p> <p>Line beginning with: Cat::Cat should not have empty body</p> <p>Line b = s; does not compile. Type mismatch</p> <p>Line Cat b; does not compile. No suitable constructor.</p> <p>Line Cat b; does not link. No suitable implementation.</p>	<p>Line Cat b; does not link. No suitable implementation.</p>
<p>[2124] What happens with this code?</p> <pre>#include <string> #include <iostream> using namespace std; class Cat { string name_; public: Cat() = default; explicit Cat(const string& n); string get() const; }; Cat::Cat(const string& n): name_(n) {} string Cat::get() const { return name_; }</pre> <p>int main()</p> <p>{</p> <p>string s = "Bill";</p> <p>Cat b;</p> <p>b = s;</p> <p>cout << b.get() << endl;</p> <p>}</p> <p>Line beginning with: Cat::Cat should not have empty body</p> <p>The does compile; it prints "Bill".</p> <p>Line Cat b; does not compile. No suitable constructor.</p> <p>Line Cat b; does not link. No suitable implementation.</p> <p>Line b = s; does not compile. Type mismatch</p>	<p>Line b = s; does not compile. Type mismatch</p>



<pre>#include <string> #include <iostream> using namespace std; class Cat { string name_; public: Cat() = default; explicit Cat(const string& n); string get() const; }; explicit Cat::Cat(const string& n): name_(n) {} string Cat::get() const { return name_; }</pre> <p>int main()</p> <p>{</p> <p>string s = "Bill";</p> <p>Cat b;</p> <p>b = s;</p> <p>cout << b.get() << endl;</p> <p>}</p> <p>Line beginning with: explicit Cat::Cat should not repeat explicit in implementation</p> <p>Line beginning with: string Cat::get should not repeat const in implementation</p> <p>Line Cat b; does not compile. No suitable constructor.</p> <p>Line b = s; does not compile. Type mismatch</p> <p>The does compile; it prints "Bill".</p>	
<p>[2126] What happens with this code?</p> <pre>#include <string> #include <iostream> using namespace std; class Cat { string name_; public: Cat() = default; explicit Cat(const string& n); string get() const; }; Cat::Cat(const string& n): name_(n) {} string Cat::get() { return name_; }</pre> <p>int main()</p> <p>{</p> <p>string s = "Bill";</p> <p>Cat b;</p> <p>b = s;</p> <p>cout << b.get() << endl;</p> <p>}</p> <p>Line beginning with: string Cat::get should repeat const in implementation</p> <p>Line b = s; does not compile. Type mismatch</p> <p>The does compile; it prints "Bill".</p> <p>Line Cat b; does not compile. No suitable constructor.</p> <p>Line beginning with: Cat::Cat should not have an empty body.</p>	<p>Line beginning with: string Cat::get should repeat const in implementation</p>
<p>A behavior of an object is represented by the messages that it responds to.</p>	<p>True</p>
<p>Using encapsulation such as that used with structures, risks accidental data corruption.</p>	<p>False</p>
<p>Constructors always have the same name as the class, except that the constructor name is capitalized.</p>	<p>False</p>
<p>In C++11 you can initialize members in the initializer list using braces, parentheses or the assignment operator syntax.</p>	<p>False</p>
<p>Inheritance enforces the principle of data hiding.</p>	<p>False</p>
<p>Constructors must be explicitly called after an object is created.</p>	<p>False</p>
<p>Constructors are called implicitly whenever an object is created.</p>	<p>True</p>
<p>In a constructor, initializing data members by using the assignment operator means that those objects may be initialized twice.</p>	<p>True</p>
<p>The constructor that is used to initialize all of an object's fields is called the working constructor.</p>	<p>True</p>
<p>Constructors always have the same name as the class and a return type of void.</p>	<p>False</p>
<p>Initialization of data members occurs according to the order they are listed in the class definition.</p>	<p>True</p>



The state of an object refers to the names and types of its data members.	False
With inheritance, the class you build upon is called a base class in C++.	True
With inheritance, the class you build upon is called a superclass in C++.	False
Suppose you have two classes related by inheritance: Dog and Poodle. According to the principle of substitutability, a function void walk(Poodle& p) will accept a Dog as an argument.	False
The state of an object refers to the combination of values stored in its data members.	True
Creating objects from a class is called instantiation.	True
If you do not create a constructor for your class, C++ will synthesize a working constructor for you.	False
The attributes of an object refers to the names and types of its data members.	True
If you do not create a constructor for your class, C++ will synthesize a default constructor for you.	True
A constructor is a member function whose job is to initialize an object into a well-formed state.	True
A constructor that takes a single argument of a different type is also known as a conversion constructor.	True
Polymorphism enforces the principle of data hiding.	False
A constructor that takes a single argument of a different type may be called implicitly.	True
Constructors always have the same name as the class and no return type.	True
A class specifies the behavior of the objects it creates through the definition of embedded functions, called member functions.	True
The constructor that is used to initialize all of an object's fields is called the default constructor.	False
With inheritance, the new class you create is called a base class in C++.	False
A function that is marked with the keyword inline should be placed in the implementation .cpp file.	False
Object behavior is implemented by data member.	False
Constructors always have the same name as the class, preceded by the tilde character (~).	False
A class represents a template or blueprint for creating objects of a particular kind.	True
Polymorphism only works in the presence of inheritance.	True
Your class may have more than one constructor.	True
Encapsulation enforces the principle of data hiding.	True
An object (in the OOP sense) is an instance of a particular class.	True
Object behavior is implemented by member functions.	True
A function that is marked with the keyword inline must be places in the header file.	True
C++11 you can ask the compiler to retain the synthesized constructor when adding new ones.	True
With inheritance, the class you build upon is called a derived class in C++.	False
The constructor that takes no arguments is called the working constructor.	False
Although not possible in earlier versions of C++, in C++11 you can ask the compiler to retain the synthesized constructor when adding new ones.	True
With inheritance, the new class you create is called a derived class in C++.	True
Objects are variables of programmer-defined types.	True
Marking a constructor with the explicit keyword, prevents unintended conversions.	True



With inheritance, the new class you create is called a subclass in C++.	False
In a constructor, objects can be initialized immediately before the opening brace of the constructor, before any other code has been run.	True
In a constructor, objects can be initialized immediately after the opening brace of the constructor, before any other code has been run.	False
With inheritance, a family of related classes is called a class hierarchy.	True
A reference variable has the same identity as the variable it refers to.	True
The constructor that takes no arguments is called the default constructor.	True
A reference variable has a different identity than the variable it refers to.	False
Suppose you have two classes related by inheritance: Dog and Poodle. According to the rules of inheritance, Poodle is a specialization of Dog.	True
The constructor initializer list is preceded by a colon and followed by a semicolon.	False
A class specifies the attributes of the objects it creates through the definition of internal data members.	True
Polymorphism means that different objects (of different types) can respond to the same message in different ways.	True
Suppose you have two classes related by inheritance: Dog and Poodle. According to the rules of inheritance, Dog is a specialization of Poodle.	False
With inheritance, the class you build upon is called a subclass in C++.	False
The attributes of an object refers to the combination of values stored in its data members.	False
The constructor initializer list follows the parameter list and precedes the constructor body.	True
Suppose you have two classes related by inheritance: Dog and Poodle. According to the principle of substitutability, a function void walk(Dog& d) will accept a Poodle as an argument.	False
When not using encapsulation, such with structures, you risk accidental data corruption.	True
Initialization of data members occurs according to the order they are listed in the initializer list.	False
To ask a particular object to perform a particular action, you send your request by calling a member function.	True
When not using encapsulation, such with structures, changing the implementation of the structure changes the interface as well.	True
With inheritance, the new class you create is called a superclass in C++.	False
A class specifies the attributes of the objects it creates through the definition of embedded functions, called member functions.	False
<p>[2201] The BigNum class allows you to create arbitrarily large numbers, without any approximations. Assume you have the following code. What is the best header for the required operator?</p> <pre>BigNum a{"12345.795"}, b{"-.95873421"}; auto c = a * b; const BigNum operator*(const BigNum& lhs, const BigNum& rhs); BigNum BigNum::operator*(const BigNum& rhs) const; BigNum& operator*(const BigNum& lhs, const BigNum& rhs); BigNum operator*(const BigNum& lhs, const BigNum& rhs);</pre>	<pre>const BigNum operator*(const BigNum& lhs, const BigNum& rhs);</pre>

Total combine1		Study	
<p>assuming that the BigNum constructor is non-explicit?</p> <pre>BigNum a{9.2573e27}; auto c = 100.0 / a;</pre> <p>==</p> <pre>const BigNum BigNum::operator/(const BigNum& rhs) const;</pre> <p>const BigNum operator/(const BigNum& lhs, const BigNum& rhs);</p> <p>const BigNum operator/(double lhs, const BigNum& rhs);</p> <p>All of these can be used</p> <p>==</p>			
<p>[2203] The BigNum class allows you to create arbitrarily large numbers, without loss of precision. Which of the following operators (which are all valid) cannot be used, assuming that the BigNum constructor is non-explicit?</p> <pre>BigNum a{9.2573e27}; auto c = a / 100.0;</pre> <p>==</p> <pre>const BigNum BigNum::operator/(const BigNum& rhs)const;</pre> <p>const BigNum operator/(const BigNum& lhs, const BigNum& rhs);</p> <p>const BigNum operator/(const BigNum& lhs, double rhs);</p> <p>All of these can be used</p> <p>==</p>		All of these can be used	
<p>[2204] The BigNum class allows you to create arbitrarily large numbers, without loss of precision. Examine the code shown. Which expression invokes the operator defined here?</p> <pre>BigNum a{"12345.795"}, b{".95873421"}; const BigNum BigNum::operator-(const BigNum& n) const {...}</pre> <p>==</p> <pre>a -= b;</pre> <p>auto c = a - b;</p> <p>auto c = --b;</p> <p>None of these</p> <p>auto c = -b;</p> <p>==</p>		auto c = a - b;	
<p>[2205] The BigNum class allows you to create arbitrarily large numbers, without loss of precision. Examine the code shown. Which expression invokes the operator defined here?</p> <pre>BigNum a{"12345.795"}, b{".95873421"}; const BigNum operator-(const BigNum& n) {...}</pre> <p>==</p> <p>None of these</p> <p>auto c = a - b;</p> <p>a -= b;</p> <p>auto c = --b;</p> <p>auto c = -b;</p> <p>==</p>		auto c = -b;	
<p>[2206] The BigNum class allows you to create arbitrarily large numbers, without loss of precision. Examine the code shown. Which expression invokes the operator defined here?</p> <pre>BigNum a{"12345.795"}, b{".95873421"}; const BigNum operator-() {...} auto c = a - b; a -= b; auto c = -b; auto c = --b; None of these</pre>		None of these	

Total combine1		Study	
<div>defined here?</div> <div>BigNum a{"12345.795"}, b{"95873421"}; const BigNum operator-(const BigNum&, const BigNum&) {...}</div> <div>==</div> <div>None of these</div> <div>a -= b;</div> <div>auto c = a - b;</div> <div>auto c = --b;</div> <div>auto c = -b; ==</div>			
<div>[2208] The Date class represents a day on a calendar. Examine the code shown. Which operator is called?</div> <div>Date d{2018, 7, 4}; auto e = d++; const Date Date::operator++(int); const Date Date::operator++(); Date& Date::operator++(); Date& Date::operator++(int);</div>	<div>const Date Date::operator++(int);</div>		
<div>[2209] The Date class represents a day on a calendar. Examine the code shown. Which operator is called?</div> <div>Date d{2018, 7, 4}; auto e = d++; Date& operator++(Date&, int); None of these const Date operator++(Date&, int); const Date operator++(Date&); Date& operator++(Date&);</div>	<div>const Date operator++(Date&, int);</div>		
<div>[2210] The Date class represents a day on a calendar. Examine the code shown. Which operator is called?</div> <div>Date d{2018, 7, 4}; auto e = ++d; const Date Date::operator++(int); Date& Date::operator++(int); None of these const Date Date::operator++(); Date& Date::operator++();</div>	<div>Date& Date::operator++();</div>		
<div>[2211] The Date class represents a day on a calendar. Examine the code shown. Which operator is called?</div> <div>Date d{2018, 7, 4}; auto e = ++d; == const Date operator++(Date&); Date& operator++(Date&, int); Date& operator++(Date&); const Date operator++(Date&, int); None of these ==</div>	<div>Date& operator++(Date&);</div>		
<div>[2214] The Time class represents the time of day on a clock. Examine the code shown. Which operator is called?</div> <div>Time t(8, 30, "a"); cout << t << endl; == ostream& ostream::operator<<(const Time&); ostream operator<<(ostream, Time); ostream& operator<<(ostream&, const Time&); None of these ostream&Time::operator<<(ostream&, const Time&); ==</div>	<div>ostream& operator<<(ostream&, const Time&);</div>		

Total combine1		Study	
<pre>class Point { int x_{0}, y_{0}; public: int x() const; int y() const; }; bool operator==(const Point& lhs, const Point& rhs) { return _____; } this->x() == rhs.() && this->y() == rhs.y() None of these lhs.x() == rhs.() && lhs.y() == rhs.y() lhs.x_ == rhs.x_ && lhs.y_ == rhs.y_ this->x_ == rhs.x && this->y_ == rhs.y_</pre>			
<p>[2216] The Point class represents x,y coordinates in a Cartesian plane. Which line of code appears in the blank? (Members written inline for this problem.)</p> <pre>class Point { int x_{0}, y_{0}; public: int x() const { return x_; } int y() const { return y_; } bool operator==(const Point& rhs) const { return _____; } }; x_ == rhs.x_ && y_ == rhs.y_ x() == rhs.() && y() == rhs.y() this->x() == rhs.() && this->y() == rhs.y() this->x_ == rhs.x && this->y_ == rhs.y_ All of these will work</pre>		All of these will work	
<p>[2217] The Point class represents x,y coordinates in a Cartesian plane. Which line of code appears completes this operator which transforms a Point by dx and dy? (Members written inline for this problem.)</p> <pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } Point operator+(int dx, int dy) const { return _____; } }; Point(x() + dx, y() + dy); Point(x_ + y_, dx + dy); Does not compile, changes arity of operator. Does not compile; must be a non-member function. Does not compile; must have one user-defined type as argument.</pre>		Does not compile, changes arity of operator.	
<p>[2218] The Point class represents x,y coordinates in a Cartesian plane. Which line of code appears completes this operator which transforms a Point by dx and dy? (Members written inline for this problem.)</p> <pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } }; Point operator+(int dx, int dy) { return _____; } == Does not compile; must have one user-defined type as argument. Does not compile, changes arity of operator Does not compile; must be a member function Point(x_ + y_, dx + dy) Point(x() + dx, y() + dy) =</pre>		Does not compile; must have one user-defined type as argument.	



<p>in memory? (Members written inline for this problem.)</p> <pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } }; Point operator@(const Point& p) { return _____; } Does not compile; uses a non-operator symbol. Does not compile; changes arity of operator. Does not compile; must be a member function. *&p &p</pre>	
<p>[2220] The Point class represents x,y coordinates in a Cartesian plane. Which line of code appears completes this operator? (Members written inline for this problem.)</p> <pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } }; const Point operator++(Point& p, int n) { Point temp(p); ... return _____; } *this Does not compile; cannot change data members of object; no mutators. temp Does not compile; must be a member function. Does not compile; changes arity of operator; should be unary, not binary.</pre>	<p>Does not compile; cannot change data members of object; no mutators.</p>
<p>[2221] The Point class represents x,y coordinates in a Cartesian plane. Which line of code appears completes this operator? (Members written inline for this problem.)</p> <pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } const Point operator++(int n) { Point temp(*this); ... return _____; } }; temp *this Does not compile; changes arity of operator; should be unary, not binary. Does not compile; must be a non-member function. Does not compile; cannot change data members of object; no mutators.</pre>	<p>temp</p>
<p>[2222] The Point class represents x,y coordinates in a Cartesian plane. Which line of code appears completes this operator? (Members written inline for this problem.)</p> <pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } Point& operator++() { Point temp(*this); ... return _____; } }; *this temp Does not compile; cannot change data members of object; no mutators Does not compile; must be a non-member function Does not compile; changes arity of operator; should be unary, not empty</pre>	<p>*this</p>



<pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } }; Point& operator++(Point& p) { ... return _____; } Does not compile; cannot change data members of object; no mutators. Does not compile; changes arity of operator; should be unary, not empty. p Does not compile; must be a non-member function. *this</pre>	
<p>[2224] The Point class represents x,y coordinates in a Cartesian plane. Which line of code appears completes this operator? (Members written inline for this problem.)</p> <pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } }; Point& operator+=(Point& rhs) { x_ += rhs.x(); y_ += rhs.y(); return _____; } Does not compile; missing const at the end of the operator header. *this Does not compile; changes arity of operator; should be unary, not binary. Does not compile; must be a member operator. rhs</pre>	<p>Does not compile; must be a member operator.</p>
<p>[2225] The Point class represents x,y coordinates in a Cartesian plane. Which line of code appears completes this operator? (Members written inline for this problem.)</p> <pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } }; Point& operator+=(Point& lhs, const Point& rhs) { _____ return lhs; } == lhs.x_ += rhs.x(); lhs.y_ += rhs.y(); lhs.x() += rhs.x(); lhs.y() += rhs.y(); Does not compile; rhs must not be const Does not compile; no access to private members of lhs Does not compile; changes arity of operator; should be unary, not binary ==</pre>	<p>Does not compile; no access to private members of lhs.</p>



argument. (Members written inline for this problem.)

```
class Point {
int x_{0}, y_{0};
public:
Point(int x, int y): x_{x}, y_{y} {}
int x() const { return x_; }
int y() const { return y_; }
Point& operator+=(const Point& rhs) {

return *this;
}
};
```

```
==
*this = rhs;

rhs.x_ += this->x_; rhs.y_ += this->y;
```

```
this->x() += rhs.x(); this->y() += rhs.y();
```

Does not compile; no access to private members object

```
x_ += rhs.x(); y_ += rhs.y();
==
```

[2227] The Point class represents x,y coordinates in a Cartesian plane. What is the mistake in this operator? (Members written inline for this problem.)

```
class Point {
int x_{0}, y_{0};
public:
Point(int x, int y): x_{x}, y_{y} {}
int x() const { return x_; }
int y() const { return y_; }
Point& operator+=(Point& rhs) {
...
return *this;
}
};
```

- Does not compile; should be a non-member function.
- The operator return type should be a const Point
- The parameter should be a constant reference
- The operator should end with return this, not return *this.
- The operator should have const at the end of the header

The parameter should be a constant reference

[2228] The Point class represents x,y coordinates in a Cartesian plane. What is the mistake in this operator? (Members written inline for this problem.)

```
class Point {
int x_{0}, y_{0};
public:
Point(int x, int y): x_{x}, y_{y} {}
int x() const { return x_; }
int y() const { return y_; }
Point operator+=(const Point& rhs) {
...
return *this;
}
};
```

- The parameter should be a non-constant reference
- The operator should have const at the end of the header
- The operator return type should be a Point&
- The operator should end with return this, not return *this

```
Does not compile; should be a non-member function
==
```

The operator return type should be a Point&



<pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } Point& operator+=(const Point& rhs) { . . . } }; const Point operator+(Point& lhs, const Point& rhs) { return lhs += rhs; } The operator should not change any of its parameters There is no error; it works fine. Does not compile; should be a member function. The rhs parameter should not be const The operator should return lhs after adding rhs to it.</pre>	
<p>[2230] The Point class represents x,y coordinates in a Cartesian plane. What is the mistake in this operator? (Members written inline for this problem.)</p> <pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } Point& operator+=(const Point& rhs) { . . . } }; const Point operator+(const Point& lhs, const Point& rhs) { return Point(lhs) += rhs; } The operator should not change any of its parameters The operator return type should be a Point&. The rhs parameter should not be const Does not compile; should be a member function. There is no error; it works fine.</pre>	<p>There is no error; it works fine.</p>
<p>[2231] The Point class represents x,y coordinates in a Cartesian plane. What is the mistake in this operator? (Members written inline for this problem.)</p> <pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } Point& operator+=(const Point& rhs) { . . . } }; Point& operator+(const Point& lhs, const Point& rhs) { return Point(lhs) += rhs; } == There is no error; it works fine. Does not compile; should be a member function The rhs parameter should not be const The operator should not change any of its parameters The operator return type should not be a Point&. ==</pre>	<p>The operator return type should not be a Point&.</p>



```
class Point {
int x_{0}, y_{0};
public:
Point(int x, int y): x_{x}, y_{y} {}
int x() const { return x_; }
int y() const { return y_; }
};

ostream& operator<<(ostream& out, const Point& p)
{
return out << '(' << p.x() << ", " << p.y() << ')';
}

==
There is no error; it works fine

The Point p parameter should not be const

Does not compile; should be a member function

You must return out after writing to it. This example returns void

The data members x_ and y_ are inaccessible in a non-member function
==
```

[2233] The Point class represents x,y coordinates in a Cartesian plane. What is the mistake in this operator? (Members written inline for this problem.)

```
class Point {
int x_{0}, y_{0};
public:
Point(int x, int y): x_{x}, y_{y} {}
int x() const { return x_; }
int y() const { return y_; }
};

ostream& operator<<(ostream& out, const Point& p)
{
return out << '(' << p.x_ << ", " << p.y_ << ')';
}

==
The Point p parameter should not be const

Does not compile; should be a member function

There is no error; it works fine.

You must return out after writing to it. This example returns void

The data members x_ and y_ are inaccessible in a non-member function
==
```

The data members x_ and y_ are inaccessible in a non-member function

You must return out after writing to it. This example returns void.

Total combine1		Study	<div></div>
<div><pre>class Point { int x_{0}, y_{0}; public: Point(int x, int y): x_{x}, y_{y} {} int x() const { return x_; } int y() const { return y_; } }; ostream& operator<<(ostream& out, Point& p) { return out << '(' << p.x() << ", " << p.y() << ')'; } The data members x_ and y_ are inaccessible in a non-member function. There is no error; it works fine. The Point p parameter should be const Does not compile; should be a member function. You must first write to out and then return it.</pre></div>			
<div><div>The prototype for a member subtraction operator for the type T is:</div><div>const T operator-(const T& lhs, const T& rhs);</div></div>		False	
<div><div>The prototype for a non-member addition operator for the type T is:</div><div>const T operator+(const T& rhs) const;</div></div>		False	
<div><div>The prototype for a non-member subtraction operator for the type T is:</div><div>const T operator-(const T& lhs, const T& rhs);</div></div>		True	
<div><div>You may not overload the scope operator ::.</div></div>		True	
<div><div>The parameter names lhs and rhs are commonly used with overloaded operators.</div></div>		True	
<div><div>Overloaded operators may be implemented as member functions.</div></div>		True	
<div><div>The expression *this can be returned from non-member operators.</div></div>		False	
<div><div>The short-hand assignment operators for type T should return *this.</div></div>		True	
<div><div>The expression *this can only be returned from member operators.</div></div>		True	
<div><div>With operator overloading, you may use any symbol to define a new operator.</div></div>		False	
<div><div>You must use the ordinary meaning of an operator when you overload it. It would be impossible to redefine subtraction to mean addition, for instance.</div></div>		False	
<div><div>The prototype for a member addition operator for the type T is:</div><div>const T operator+(const T& rhs) const;</div></div>		True	
<div><div>To compare objects for equality, overload both == and !=.</div></div>		True	
<div><div>Though not required, you should use the ordinary meaning of an operator when you overload it. It would be unwise to redefine subtraction to mean addition, for instance.</div></div>		True	
<div><div>Overloaded operators are functions that use special names that begin with the keyword overloaded.</div></div>		False	
<div><div>The arithmetic operators, such as addition, subtraction and multiplication for type T should return a const T.</div></div>		True	
<div><div>The signature for the postfix decrement operator (of type T) is:</div><div>T& operator--();</div></div>		False	
<div><div>You may overload the conditional operator ?..</div></div>		False	
<div><div>The expression *this is called a self-reference.</div></div>		True	
<div><div>Classes whose objects need to be sorted should overload == and !=.</div></div>		True	
<div><div>You can only overload existing operators. You cannot use other symbols.</div></div>		True	
<div><div>The signature for the postfix decrement operator (of type T) is:</div><div>const T operator--(int);</div></div>		True	
<div><div>The subscript operators must be written as a member operator.</div></div>		True	
<div><div>You may overload operators for the built-in types.</div></div>		False	



The I/O operators must always be written as non-member operators.	True
Symetric operators, where the user-defined type may appear on the left or the right, should be written as member operators.	False
The short-hand assignment operators for type T should return a T&.	True
Side-effect operators, such as increment or short-hand assignment, should be written as member operators.	True
The signature for the prefix increment operator (of type T) is: const T operator++(int);	False
You may not overload the subscript ([]) operator.	False
Symetric operators, where the user-defined type may appear on the left or the right, should be written as non-member operators.	True
The arithmetic operators, such as addition, subtraction and multiplication for type T should return a T.	False
Overloaded operators may be implemented as non-member functions.	True
An overloaded operator must have at least one operand that is a user-defined type.	True
The short-hand assignment operators for type T should return a const T.	False
Member operators have direct access to private data members.	True
The I/O operators should always be written as member operators.	False
You may not overload the conditional operator ?:	True
The parameter names left and right are commonly used with overloaded operators.	False
Classes whose objects need to be sorted should overload <.	False
The subscript operators may be written as a member operator or as a non-member operator.	False
Side-effect operators, such as increment or short-hand assignment, should be written as non-member operators.	False
Non-member operators have direct access to private data members.	True
The arithmetic operators, such as addition, subtraction and multiplication for type T should return a T&.	False
You may not overload the indirection operator, the unary *.	False
[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



<pre>int mystery(int n) { if (n == 1) return 1; return n * mystery(n-1); }</pre> <p>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</p>	
<p>[2304] What does this function do?</p> <pre>int mystery(int n) { if (n < 2) return 1; return mystery(n-1) + mystery(n-2); }</pre> <p>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</p>	Computes the Fibonacci number n
<p>[2305] What does this function do?</p> <pre>int mystery(int n) { if (n == 1) return 1; return n * mystery(n+1); }</pre> <p>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</p>	Produces a stack overflow
<p>[2306] What does this function do?</p> <pre>int mystery(int n) { if (n == 1) return 1; return n * mystery(n-1); }</pre> <p>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</p>	Computes the Gauss series (sum) of 1..n
<p>[2307] What does this function do?</p> <pre>int mystery(int n, int m) { if (n == 0) return m; return m * 10 + mystery(n / 10) + n % 10; }</pre> <p>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</p>	Computes the reverse of the input n
<p>[2308] What is the value of mystery(12)?</p> <pre>int mystery(int n) { if (!n) return 0; return 2 * mystery(n-1); }</pre> <p>18 24 36 12</p>	24




<pre>int r(int n) { if (n > 0) return n + r(n - 1); return n; }</pre> <p>15 6 10 24 21</p>	
<p>[2310] What is the value of mystery(5)?</p> <pre>int mystery(int n) { if (n > 0) return 3 - n % 2 + mystery(n-1); return 0; }</pre> <p>7 12 5 10 15</p>	12
<p>[2311] What is the value of r(126)?</p> <pre>int r(int n) { if (n >= 10) return n % 10 + r(n / 10); return n; }</pre> <p>3 6 13 10 9</p>	9
<p>[2312] What is the value of r(12777)?</p> <pre>int r(int n) { if (0 == n) return 0; int x = n % 10 == 7; // 0 or 1 return x + r(n / 10); }</pre> <p>5 Does not compile 2 3 Stack overflow</p>	3
<p>[2313] What is the value of r(74757677)?</p> <pre>int r(int n) { if (n) return (n % 10 == 7) + r(n / 10); return 0; }</pre> <p>3 5 Does not compile 8 Stack overflow</p>	5
<p>[2314] What is the value of r(74757677)?</p> <pre>int r(int n) { if (n) return (n % 10 != 7) + r(n / 10); return 0; }</pre> <p>5 3 Does not compile 8 Stack overflow</p>	3

Total combine1		Study	
<pre>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</pre>			
<p>[2316] What is the value of r(81238)?</p> <pre>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</pre>	2		
<p>[2317] What is the value of r(88788)?</p> <pre>int r(int n) { if (!n) return 0; return (n % 10 == 8) + (n % 100 == 88) + r(n / 10); } 4 1 5 6 Stack overflow</pre>	6		
<p>[2318] What is the value of r(3, 3)?</p> <pre>int r(int n, int m) { if (m) return n * r(n, m - 1); return 1; } 12 27 Stack overflow 9 3</pre>	27		
<p>[2319] What is the value of r("xxhixx")?</p> <pre>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</pre>	4		
<p>[2321] What is the value of r("xxhixx")?</p> <pre>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</pre>	yyhiyy		



<pre>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 xyxyxyx yhiyhiy xyhixyhixy</pre>	
<p>[2323] What is the value of r("axxbxx")?</p> <pre>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</pre>	"ab"
<p>[2324] What is the value of r("axxbxx")?</p> <pre>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"</pre>	"xxxx"
<p>[2325] Assume you have the array: int a[] = {1, 11, 3, 11, 11};. What is the value of r(a, 0, 5)?</p> <pre>int r(const int a[], size_t i, size_t max) { if (i < max) return (a[i] == 11) + r(a, i + 1); return 0; }</pre> <p>3 5 Stack overflow 1 0</p>	3
<p>[2326] What is the value of r("hello")?</p> <pre>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"</pre>	"hello"

Total combine1		Study	<div>...</div>
<pre>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; }</pre> <p>"hello" Stack overflow "hel*o" "hello" "hel*lo"</p>			
<p>[2328] What is the value of r("hello")?</p> <pre>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; }</pre> <p>"hel*o" "hel*lo" "hello" Stack overflow "hello"</p>		"h*e*ll*o"	
<p>[2329] What is the value of r("hello")?</p> <pre>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; }</pre> <p>"hello" Stack overflow "hel*o" "hel*lo"</p> <p>"*h*el*lo"</p>		"*h*el*lo"	
<p>[2330] Which of the following statements is correct about a recursive function?</p> <p>A recursive function must never call another function.</p> <p>A recursive function calls itself.</p> <p>A recursive function must be simple.</p> <p>A recursive function must call another function.</p>		A recursive function calls itself.	
<p>[2331] What does this function do?</p> <pre>void myfun(string word) { if (word.length() == 0) return; myfun(word.substr(1, word.length())); cout << word[0]; }</pre> <p>Prints the length of the string word</p> <p>Prints the string word both forward and reverse</p> <p>Prints the string word in reverse</p> <p>Prints the string word</p>		Prints the string word in reverse	
<p>[2332] What changes about this function if lines 4 and 5 are swapped?</p> <pre>1. void myfun(string word) 2. { 3. if (word.length() == 0) { return; } 4. myfun(word.substr(1, word.length())); 5. cout << word[0]; 6. }</pre> <p>prints the characters of the string in both forward and reverse order</p> <p>creates infinite recursion</p> <p>nothing</p> <p>reverses the order in which the characters of the string are printed</p>		reverses the order in which the characters of the string are printed	

Total combine1		Study	
<p>Recursion always helps you create a more efficient solution than other techniques.</p> <p>A recursion eventually exhausts all available memory, causing the program to terminate</p> <p>A recursive computation solves a problem by calling itself with simpler input.</p> <p>None of the listed options.</p>			
<p>[2334] How can you ensure that a recursive function terminates?</p> <p>Call the recursive function with simpler inputs.</p> <p>Use more than one return statement.</p> <p>Provide a special case for the simplest inputs.</p> <p>Provide a special case for the most complex inputs.</p>		Provide a special case for the simplest inputs	
<p>[2335] Which of the following is a key requirement to ensure that recursion is successful?</p> <p>Every recursive call must simplify the computation in some way</p> <p>A recursive solution should not be implemented to a problem that can be solved iteratively</p> <p>There should be special cases to handle the most complex computations directly</p> <p>A recursive function should not call itself except for the simplest inputs</p>		Every recursive call must simplify the computation in some way.	
<p>[2336] What is the value of r(3)?</p> <pre>int r(int n) { if (n < 2) { return 1; } return n * r(n - 1); }</pre> <p>24</p> <p>2</p> <p>120</p> <p>6</p>		6	
<p>[2337] Which statement ensures that r() terminates for all values of n?</p> <pre>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; }</pre>		if (n < 1) { return 1; }	
<p>[2338] Infinite recursion can lead to an error known as</p> <p>stack overflow</p> <p>heap exhaustion</p> <p>heap fragmentation</p> <p>memory exception</p>		stack overflow	
<p>[2339] Infinite recursion can occur because</p> <p>the base case is missing one of the necessary termination conditions</p> <p>the recursive function is called more than once</p> <p>the recursive case is invoked with simpler arguments</p> <p>a second function is called from the recursive one</p>		the base case is missing one of the necessary termination conditions	
<p>[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?</p> <pre>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;}</pre>		if (number <= 1) { return 1.0;}	

Total combine1		Study	<div>100%</div>
<p>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</p> <p>If the function double golden (int) is a recursive implementation of this function, what should be the recursive call in that function?</p> <pre>return sqrt (1.0 + golden(number)); return sqrt (1.0 + golden(number - 1)); return (1.0 + golden(number - 1)); return (1.0 + golden(number));</pre>			
<p>[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?</p> <p>Yes, but the code will be very long</p> <p>No, because the base case is not zero</p> <p>Yes</p> <p>No, because there is no base case</p>		Yes	
<p>[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</p> <p>The formula states that equal to the limit, as n goes to infinity, of the series</p> <p>Which function below is a correct recursive implementation that approximates this infinite series?</p>		<pre>double computePI(int number) { if (number <= 1) { return 1.0;} return 1.0 / (number * number) + computePI(number - 1); }</pre>	
<p>[2344] 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 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?</p> <p>if (number == 0) { return 1.0 / (number * number);}</p> <p>if (number <= 1) { return 1.0;}</p> <p>if (number <= 1) { return 0.0;}</p> <p>if (number == 1) { return (number * number);}</p>		if (number <= 1) { return 1.0;}	
<p>[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?</p> <pre>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);</pre>		return 1.0 / (number * number) + computePI(number - 1);	
<p>[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:</p> <pre>double computePI(int number) { if (number <= 1) { return 1.0;} int oddnum = 2 * number - 1; return computesign(number) * 1.0 / oddnum + computePI(number - 1); }</pre> <p>In this recursive function, what is the recursive base case?</p> <p>When the parameter variable is less than or equal to one</p> <p>When the parameter variable is greater than one</p> <p>When the value that is returned from the function is zero</p> <p>When the parameter variable is zero</p>		When the parameter variable is less than or equal to one	

Total combine1		Study	<div>100%</div>
<p>recursive function below to calculate this formula:</p> <pre>double computePI(int number) { if (number <= 1) { return 1.0;} int oddnum = 2 * number - 1; return computesign(number) * 1.0 / oddnum + computePI(number - 1); }</pre>			
<p>In this recursive function, what is the role of the helper function computesign?</p> <p>it is the recursive call in the function</p> <p>it checks the sign of the number and returns true if it is positive and false if negative</p> <p>it is called just one time to set the sign of the final result</p> <p>it makes sure the sign (positive or negative) alternates as each term of the series is computed</p>			
<p>[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?</p> <p>Call calc_prod(n - 1) and multiply by n.</p> <p>Call calc_prod(n + 1) and multiply by n.</p> <p>Call calc_prod(n - 2) and multiply by n.</p> <p>Call calc_prod(1) and multiply by n.</p>		Call calc_prod(n - 1) and multiply by n.	
<p>[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?</p> <p>Call power(x, n) and multiply by (n - 1).</p> <p>Call power(x, n - 1) and multiply by n.</p> <p>Call power(x - 1, n) and multiply by x.</p> <p>Call power(x, n - 1) and multiply by x.</p>		Call power(x, n - 1) and multiply by x.	
<p>[1] What must I change in the test to go to the next iteration?</p> <p>[2] What information is produced?</p> <p>[3] What must I do to enter the loop?</p> <p>[4] Can my loop reach its bounds?</p> <p>[5] Has my loop reached its goal?</p> <p>[6] How is the data processed?</p> <p>[7] Can my loop be entered at all?</p> <p>[8] What makes this loop quit?</p>		<p>[1] advance the loop</p> <p>[2] goal precondition</p> <p>[3] bounds precondition</p> <p>[4] necessary bounds</p> <p>[5] loop postcondition</p> <p>[6] loop operations and actions</p> <p>[7] loop guards</p> <p>[8] loop bounds</p>	
<p>[1] May not repeat its actions at all</p> <p>[2] Keeps processing input until a particular value is found in input.</p> <p>[3] Repeats its actions at least once</p> <p>[4] Keeps processing until the output gets no closer to the answer.</p> <p>[5] Test for the occurrence of a particular event</p> <p>[6] Repeats its actions a fixed number of times</p> <p>[7] Conditions under which a loop will repeat its actions</p> <p>[8] Keeps processing until the input device signals that it is finished.</p>		<p>[1] guarded loop</p> <p>[2] sentinel loop</p> <p>[3] unguarded loop</p> <p>[4] limit loop</p> <p>[5] indefinite loop</p> <p>[6] definite loop</p> <p>[7] loop bounds</p> <p>[8] data loop</p>	
<p>[1] Actions that occur after the loop is complete</p> <p>[2] Actions occurring inside the loop's body</p> <p>[3] Actions that occur before the loop is encountered</p> <p>[4] A test that determines if the loop should be entered</p>		<p>[1] postcondition</p> <p>[2] operation</p> <p>[3] precondition</p> <p>[4] bounds</p>	
<p>Which of these is a flow-of-control statement?</p>		<p>for (auto e : s) ...</p> <p>if (x < 3) ... else ...</p> <p>while (x < 3) ...</p>	
<p>Which of these are guarded loops?</p>		<p>for</p> <p>while</p>	
<p>Which of these are unguarded loops?</p>		<p>do-while</p>	
<p>Which are the two major categories of loops?</p>		<p>definite</p> <p>indefinite</p>	
<p>Which of these are indefinite loops?</p>		<p>sentinel bounds</p> <p>limit bounds</p> <p>data bounds</p>	
<p>Using the loop-building strategy from Chapter 5, which of these are part of the loop mechanics?</p>		<p>loop bounds</p> <p>bounds precondition</p> <p>advancing the loop</p>	



[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:	asymmetric 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



<p>Given: the variable str is a string (may be empty) Create the counter variable, initialized to -1</p> <p>If the variable str has any characters then {</p> <p>Set counter to 0 Create the variable current-character as a character Place the first character in str into current-character</p> <p>While more-characters and current-character not a period { Add one to (or increment) the counter variable</p> <p>☀ Store the next character from str in current-character ☀</p> <p>}</p> <p>If current-character is a period then Add one to the counter to account for the period. Else Set counter to -2</p> <p>}</p> <p>If counter is -1 the string was empty Else if counter is -2 there was no period</p>	
<p>Below is the illustration from the loop building strategy in Chapter 5. The highlighted lines represents:</p> <p>Given: the variable str is a string (may be empty) Create the counter variable, initialized to -1</p> <p>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</p> <p>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 }</p> <p>☀ If current-character is a period then ☀</p> <p>Add one to the counter to account for the period. Else Set counter to -2]</p> <p>}</p> <p>If counter is -1 the string was empty Else if counter is -2 there was no period</p>	<p>loop postcondition</p>
<p>In a guarded loop, the loop actions may never be executed</p> <p>In a guarded loop, the loop actions are always executed at least once.</p>	<p>True</p> <p>False</p>
<p>In an unguarded loop, the loop actions are always executed at least once.</p> <p>In an unguarded loop, the loop actions may never be executed.</p>	<p>True</p> <p>False</p>
<p>A guarded loop is also known as a test-at-the-top loop</p> <p>A guarded loop is also known as a test-at-the-bottom loop.</p>	<p>True</p> <p>False</p>
<p>An unguarded loop is also known as a test-at-the-bottom loop.</p> <p>An unguarded loop is also known as a test-at-the-top loop.</p>	<p>True</p> <p>False</p>
<p>Loops are used to implement iteration in C++.</p> <p>Loops are used to implement selection in C++.</p>	<p>True</p> <p>False</p>



<pre>for (int i = 1; i <= 10; i++) cout << i; cout << endl;</pre> <p>This idiomatic pattern is used to count from one value to another.</p> <pre>for (int i = 1; i < 10; i++) cout << i; cout << endl;</pre>	<p>False</p>
<p>This loop uses asymmetric bounds.</p> <pre>for (int i = 0; i < 10; i++) cout << i; cout << endl;</pre> <p>This loop uses asymmetric bounds.</p> <pre>for (int i = 1; i < 10; i++) cout << i; cout << endl;</pre> <p>This loop uses asymmetric bounds.</p> <pre>for (int i = 1; i <= 10; i++) cout << i; cout << endl;</pre>	<p>True</p> <p>True</p> <p>False</p>