Science / Computer Science

Share

CS150 Midterm 3

Leave the first rating

Terms in this set (225)

Examine the following code. Which element is erased?
vector<int> v{1, 2, 3};
v.erase(begin(v), end(v));
	All the elements are erased

When using the STL function count_if, the third argument is:	a predicate function

CS150 Midterm 3

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

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

The reset() function deletes the raw pointer that a unique_ptr contains, and then sets that pointer to a new value.

On the command line, argc is the count of arguments including the program itself.

Using a pointer to access the memory it points to after the pointer has been deleted is called a dangling pointer.

A reference variable has the same identity as the variable it refers to.

The parameter names lhs and rhs are commonly used with overloaded operators.

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.

The constructor initializer list follows the parameter list and precedes the constructor body.

In C++, an Abstract Base Class is any class that has one pure virtual member function.

The composition relationship is informally known as has-a.

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

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

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

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

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

Programmers using structure-derived variables, directly manipulate the data members of those variables.

The statement new int[3]; allocates an array of three uninitialized integers on the heap.

The statement new int{3}; allocates a single initialized integer on the heap.

It is always a logic error for a derived class to redefine a non-virtual function.

The iostream class in the C++ standard library uses multiple inheritance.

You may create a reference to a class that is abstract.

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

Conceptually, a 2D array is rectangular grid of columns and rows.

Arrays can have more than two dimensions in C++.

If the new operator cannot allocate memory, C++ throws an exception.

You may not overload the conditional operator ?:.

A unique_ptr may transfer its ownership to another unique_ptr.

Consider the Shape class hierarchy, along with Circle, Square and Star from your text. The Shape class is an abstract base class.

The strcat() function overwrites the terminating NUL in the destination string.

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

When removing an element from a partially-filled array, elements following the deletion position are shifted to the left.

The statement new int[3] = {1, 2, 3}; allocates an array of three initialized integers on the heap.

A pointer that goes out of scope before deleting the memory it points to is called a memory leak.

Accessor member functions are allowed to read data members, but not change them.

The private inheritance relationship is informally known as implemented-with.

The keyword override allows the compiler to ensure that the base-class function you are overriding is virtual.

Non-virtual functions always use early, or compile-time binding to decide which function to call.

The statement v.insert(v.end() + 1, 3) is undefined because end() + 1 points past the last element in the vector.

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

Using structures for user-defined types means that you cannot enforce restrictions on data member access.

The member function int& hours(); provides read-write access to the hours property (however it is stored).

The member function int hours() const; provides read-only access to the hours property (however it is stored).

Member functions that change the state of an object are called mutators.

With classes, the public interface includes the member functions that allow clients to access object data in a safe way.

If a member function is in the private section of a class, it can only be called by other member functions of the class.

A class is an interface paired with an implementation.

In C++ the only difference between structures and classes is that member functions are private by default in classes.

The predefined constant _cplulus indicates which version of the C++ standard is being used.

Without try and catch, the throw statement terminates the running program.

The push_back member function adds elements to the end of a vector expanding the vector's capacity if needed.

In the declaration: vector<int> v; the word int represents the object's base type.

vector subscripts begin at 0 and go up to the vector size - 1.

The size of the array is not stored along with its elements.

Array subscripts are not range checked

Building your code with more than one copy of a function leads to a clash of symbols.

To use strings as a data stream source or sink, use the <sstream> header

The standard library version of stoi("UB-40") throws a runtime exception because there is no viable conversion.

vector is a structured library type.

The clear() member function removes all the elements from a vector

A vector subscript represents the element's offset from the beginning of the vector.

Assume the vector v contains [1, 2, 3]. v.pop_back(); changes v to [1, 2].

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

The subscripts of a C++ array range from 0 to the allocated array size -1.

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.

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.

A behavior of an object is represented by the messages that it responds to.

The attributes of an object refers to the names and types of its data members.

The constructor initializer list follows the parameter list and precedes the constructor body.

A constructor that takes no arguments is called the default constructor.

You may add = default; to the prototype for a default constructor to retain the synthesized version in the presence of other overloaded constructors.

The constructor that takes no arguments is called the default constructor.

To ask a particular object to perform a particular action, you send your request by calling a member function.

A class specifies the attributes of the objects it creates through the definition of internal data members.

Creating objects from a class is called instantiation.

A constructor always has the same name as the class, and no return type.

A function that is marked with the keyword inline must be places in the header file.

If your class does not have a constructor, the compiler will synthesize a default constructor for you.

You may not overload the member-selection (or dot) operator.

Overloaded operators may be implemented as non-member functions.

You can only overload existing operators. You cannot use other symbols.

To compare objects for equality, overload both == and !=.

You may not overload the scope operator ::.

Putting the keyword final at the end of the class heading prohibits the creation of subsequent derived classes.

Waiting until runtime to determine which function to call is known as late binding.

Waiting until runtime to determine which function to call is known as dynamic dispatch.

Virtual functions invoked through a pointer to a base-class object use late binding to decide which function to callIf a virtual member function does not use the keyword final, then any derived class may override that functionIn private inheritance derived classes inherit the implementation of the base class, but not its interface.

Virtual functions invoked through a reference to a base-class object use late binding to decide which function to call.

Virtual member functions are implemented by adding a new pointer to every object that contains at least one virtual functionIn private inheritance a using declaration is employed to selectively bring base class members into the derived class scope.

Tell the compiler that you intend to override a base class function by adding the keyword override to the end of the member function declaration.

Putting the keyword final at the end of a virtual member function heading prohibits derived classes from overriding that function.

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.

With inheritance, the new class you create is called a derived class in C++.

With inheritance, the class you build upon is called a base class in C++.

CS150 Midterm 3

Study

With inheritance, the new class you create is called a superclass in C++. Polymorphism enforces the principle of data hiding. A pointer that goes out of scope before deleting the memory it points to is called a dangling pointer. The release() function deletes the raw pointer that a unique_ptr contains, and then sets that pointer to a new value. To allocate memory on the stack, C++ uses the new operator. strcmp(s1, s2) returns a positive number if s1 is lexicographically "less than" s2. In C++, as in Java pure virtual member functions may not have an implementation. If a derived class redefine a non-virtual base-class function it causes a syntax error. If you do not create a constructor for your class, C++ will synthesize a working constructor for you. A unique_ptr uses a reference count to manage how many pointers point to an object. Requesting a block of memory from the operating system as the program runs is known as static allocation. For embedded systems, vector is preferred over arrays. 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. For an equivalent number of elements, a vector will use less memory than an array. In a 2D array the first subscript represents the columns and the second the rows. Memory for local variables is allocated on the stack when their definitions are encountered during runtime. This is known as dynamic allocation. Using a pointer to access the memory it points to after the pointer has been deleted is called a memory leak. The statement new int{}; is a syntax error. You may add = default; to the prototype of any constructor to allow the compiler to synthesize one for you. The semicolon following a class definition is optional. Since an abstract class cannot be instantiated, it is illegal to have references of abstract types. The public inheritance relationship is informally known as has-a. Putting the keyword final at the end of a non-virtual member function heading prohibits derived classes from overriding that function. In a partially-filled array, the size represents the allocated size of the array. A 2D array address expression is the equivalent of:**(address + (row** height + col)) In C++ you use the keyword public or private before each data member or member function to indicate its access privileges. Virtual functions invoked through an object use late binding to decide which function to call. The composition relationship is informally known as is-a. Virtual member functions are implemented by adding a new pointer, called a vtable, to every object that contains at least one virtual function. If you make a class final then you must make all of its member functions final as well. In private inheritance derived classes inherit the interface of the base class, but not its implementation. The public inheritance relationship is informally known as implemented-with. Non-virtual functions always use late binding to decide which function to call. Waiting until runtime to determine which function to call is known as early binding.	User-defined scalar types are created with the struct or class keywords in C++. The built-in primitive data types such as int, char and double are structured data types. User-defined types that combine multiple values into a single type are called scalar types. Using structures for user-defined types means that you can enforce restrictions on data member access. The implementation of a class normally appears entirely inside the class .cpp file. Mutator member functions are allowed to read data members, but not change them. A class definition normally appears in a .cpp file. Calling a template function like to_string<int>(3.5) is known as implicit instantiation. The pop_back member function adds elements to the end of a vector. The static_cast instruction changes way that a pointer's indirect value is interpreted. C++ arrays use bound-checking when you access their elements with the at() member function. If img is a pointer to the first byte in an image loaded into memory, Pixel is a structure, you can create a Pixel pointer pointing to the image by writing:Pixel **p = static_cast<Pixel** >(img); You may use any kind of integral variable to specify the size of a built-in C++ array. The attributes of an object refers to the combination of values stored in its data members. A reference variable has a different identity than the variable it refers to. Object behavior is implemented by data member. The constructor initializer list is preceded by a colon and followed by a semicolon. A constructor that takes no arguments is called the working constructor. A function that is marked with the keyword inline should be placed in the implementation .cpp file. The state of an object refers to the names and types of its data members. Initialization of data members occurs according to the order they are listed in the initializer list. In C++11 you can initialize members in the initializer list using braces, parentheses or the assignment operator syntax. The prototype for a member subtraction operator for the type T is: const T operator-(const T& lhs, const T& rhs); 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. With operator overloading, you may use any symbol to define a new operator. You may not overload the indirection operator, the unary *. You may overload operators for the built-in types. Overloaded operators are functions that use special names that begin with the keyword overloaded. You may overload the conditional operator ?: With inheritance, the new class you create is called a subclass in C++. Polymorphism enforces the principle of data hiding. With inheritance, the class you build upon is called a derived class in C++. With inheritance, the new class you create is called a superclass in C++. Tell the compiler that you intend to override a base class function by adding the keyword override as an annotation before the function header
Which of these is the correct syntax for a lambda which returns true if a number is evenly divisible by 4?	[](int n) { return n % 4 == 0; }
When using two iterators with an algorithm, the iterators represent an interval, or range of elements, written as:	[begin, end)

An _______ is an object which specifies the position of an element inside a container, regardless of what kind of container you use.	iterator
Assuming the following variable definition, which statement creates an object which refers to the first element in v and which prohibits you from changing v? vector<double> v{1.2, 2.3, 3.4};	auto c = cbegin(v);
The STL or Standard Template Library is a collection of data structures and algorithms developed by?	Alexander Stepanov

Examine the following code. Which element is erased?
 	
vector<int> v{1, 2, 3};
v.erase(end(v));
	Compiles but no element is erased

Assuming the following variable definition, which statement creates an object which refers to the first element in v, and which allows you to change v? vector<double> v{1.2, 2.3, 3.4};	auto a = begin(v);

Which of the following lines is legal but undefined?
 	
enum class Coin{
PENNY = 1, NICKEL = 5, DIME = 10, QUARTER = 25};
 	
Coin c;
	c = static_cast<Coin>(.25);

What prints when this code runs?
 	
enum class Coin{
PENNY = 1, NICKEL = 5, DIME = 10, QUARTER = 25};
 	
Coin c = Coin::NICKEL;
cout << static_cast<int>(c) << endl;
	5

CS150 Midterm 3

2.Expression using the dereferencing operator	2. y = *a;
3.Expression returning the number of allocated bytes used by an object	3. sizeof(Star);
4.Address value 0	4. nullptr;
5. What happens to an array when passed to a function	5. cout << a[0]; while (i < len) cout << ", " << a[i++];
6.const int * const array	6. decays
7. for (auto e : a) . .	7. Neither pointer nor elements in may be modified
8. cout << a[0]; while (i < len) cout << ", " << a[i++];	8. Fencepost algorithm
9. if (f > e) is a . . .	9.
10. if (*p == v) is a . . .	10.
11. if (v < *p) is a ...	11.
12.mystery() implements the _______ algorithm.	12. Binary Search
13. When called, mystery has infinite recursion (T/F)	13. False
14. The mystery function is recursive (T/F)	14. True
15. The mystery function has a stack overflow for certain inputs (T/F)	15. False
16. The mystery function is very efficient (T/F)	16. True
17. The variable p1 is created	17. on the stack
18. Which line allocates a variable in the static storage area	18. None of these
19. Which line uses a dangling pointer?	19. Line D
20. Which line is a double delete?	20. Line E
21. The duration of the variable a is:	21. static
22. The linkage of the variable c is:	22. none
23. The scope of the variable b is:	23. block
24. The duration of the variable d is:	24. automatic
25. Indefinite limit loop that reduces its input	25. while (n != 0) { n /= 2; }
26. Indefinite limit loop that uses successive approximations	26. while(abs(g1 - g2) >= EPSILON) {. . .}
27. Counter-controlled symmetric loop for producing a sequence of data	27. for (int i=12; i <= 19; i++) {. . .}
28. Indefinite data loop that uses raw input	28. while(cin.get(ch)) {. . .}
29. Counter-controlled asymmetric loop for processing characters	29. for (size_t i=0, len=s.size(); i < len; i++) {. . .}
30a. Iterator loop that may change its container	30a. for (auto& e : col) {. . .}
30b. Iterator loop that cannot change its container	30b. (auto e : col) {. . .}
32. Counter-controlled loop for processing substrings	32. for (size_t i=4, slen=4, len=s.size(); i < len; i++) {. . .}
33. Indefinite data loop that uses formatted input	33. while(cin >> n) {. . .}
34. Creates the vector [0]	34. vector <int> v(1);
35. Returns a reference to the fourth element in v with no range checking.	35. v[3];
36. Creates the vector [3, 3]	36. vector <int> v(2,3);
37. Adds a new element to the end of v	37. v.push_back(3);
37.	38.
38.	39.
39.	40.
40.	41.
41.	42.
42.	43.
43.	44.
44.	45.
45.	46.
46.	47.
47.	48.

49. 50.	50.
Which area of memory is your program code stored in?	Text

The value for the variable b is stored:
 	
int a = 1;
void f(int b)
{
int c = 3;
static int d = 4;
}
	On the stack

The value for the variable c is stored:
 	
int a = 1;
void f(int b)
{
int c = 3;
static int d = 4;
}
	On the stack

Which area of memory are global variables stored in?	Static storage area

The value for the variable a is stored:
 	
int a = 1;
void f(int b)
{
int c = 3;
static int d = 4;
}
	in the static storage area

All of these are legal C++ statements; which of them uses indirection? (dark in indirection) int a = 3, b = 4;	int x = *p;

What is printed when you run this code?
 	
int x(100);
cout << &x << endl;
	The memory location where x is stored

What is printed when you run this code?
 	
int *n{nullptr};
cout << *n << endl;
	No compilation errors, but undefined behavior

What is true about this code? int * choice;	choice contains an undefined address

Which line below points ppi to pi?
 	
int main()
{
double pi = 3.14159;
double *ppi;
// code goes here
// code goes here
}
	ppi = &pi;

What is true about an uninitialized pointer?	Dereferencing it is undefined behavior
What is a common pointer error?	Using a pointer without first initializing it

What is true about this code?
 	
int n{500};
int *p = &n;
	*p is the value of n

Assume that ppi correctly points to pi. Which line prints the address of ppi?	cout << ppi;

unsigned char values.) The four lines shown here are legal. Which operation is illegal?
 	
int *p1 = buf;
const int *p2 = buf;
int * const p3 = buf;
const int * p4 const = buf;	

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?
 	
int *p1 = buf;
const int *p2 = buf;
int * const p3 = buf;
const int * p4 const = buf;
	*p3 = 5;

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;

Examine this version of the swap() function. How do you call it?
 	
void swap(int * x, int & y)
{
. . .
}
. . .
int a = 3, b = 7;
// What goes here ?
	swap(&a, b);

Examine this version of the swap() function. How do you call it?
 	
void swap(int& x, int * y)
{
. . .
}
. . .
int a = 3, b = 7;
// What goes here ?
	swap(a, &b);

Examine the following code. What is stored in a after it runs.
 	
int f(int * p, int x)
{
p = x 2; return x / 2;
}
. . .
int a = 3, b, c;
c = f(&b, a);
	3

Which array definition is initialized to all zeros?
 	
int SIZE = 3;
int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[3] = {1, 2};
	a3

Which array definition produces {1, 2, 0}?
 	
int SIZE = 3;
int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[3] = {1, 2};
	a5

[1404] Which prints the number of elements in a? int a[] = {1, 2, 3};	None of these
Which line has undefined output? double speed[5] = {. . .};	cout << speed[5] << endl;
Which expression returns the number of countries? string countries[] = {"Andorra", "Albania", . . . };	None of these or sizeof(countries) / sizeof(countries[0])

int SIZE = 3
;int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[3] = {1, 2};	

Which array definition is illegal (even if it may compile on some compilers)?
 	
int SIZE = 3;
int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[3] = {1, 2};
	a1

Which assigns a value to the first position in letters? char letters[26];	letters[0] = 'a';

What is the equivalent address-offset notation?
 	
int a[] = {1, 2, 3, 4, 5, 6, 7};
int *p = a;
 	
cout << a[1] * 2 << endl;
	(p + 1) 2

What is the equivalent array notation? int dates[10]; cout << *(dates + 2) << endl; dates[2] dates[2] + 2 dates[0] + 4 &dates[2] dates[0] + 2	dates[2]

Which returns the last pixel on the first row of this image?
 	
Pixel *p; // address of pixel data
int w, h; // width and height of image
	*(p + w - 1)

What is the equivalent array notation?
 	
int dates[10];
cout << (*dates + 2) + 2 << endl;
	dates[0] + 4

What prints?
int a[] = {1, 3, 5, 7, 9};
int **p = a;cout <<** ++p;
cout << *p << endl;
 	
What prints?
int a[3][2] = {{3,2,3}};cout << a[0][0] << a[1][0] << endl;
	33
	
	Code does not compile

What is the address of the first pixel in the last row of this image?
 	
Pixel *p; // address of pixel data
int w, h; // width and height of image
	p + w * (h - 1)

What is printed here? (Assume all includes have been added. Assume 4-bytes per int, 8 bytes per pointer.)
 	
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

```
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;
}
```

What is printed?

```
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;
}
```

5

What does this function do?

```
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;
}
```

Returns the largest number in the array

What does this function do?

```
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;
}
```

Returns the largest number in the array

Below is a cumulative algorithm using an array and an iterator-based loop. What is printed? (Assume all includes have been added, etc.)

```
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;
}
```

5

```
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;
}

int main()
{
int a[] = {4, 2, 5, 2, 5, 4};
cout << mystery(a, 6) << endl;
}
```

What is printed?

```
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;
}

int main()
{
int a[] = {4, 2, 5, 2, 5, 4};
cout << mystery(a, 6) << endl;
}
```

2

Below is a cumulative algorithm using an array and an iterator-based loop. What is printed? (Assume all includes have been added, etc.)

```
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;
}
```

4

What is printed?

```
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, sizeof(a)) << endl;
```

None of these or undefined output.

What is the name for this algorithm?

```
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;
}
```

A fencepost algorithm

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?

a[size] = value;

remaining elements left, and returns true if successful?
 	
const size_t MAX = 100;
double nums[MAX];
size_t size = 0;	

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

Below is a mystery() function with no types for its parameters or local variables. What does the function do?
 	
void mystery(a, b&, d)
{
for (i = d; i < b; i++)
a[i] = a[i + 1];
b--;
}
	Deletes elements from a partially-filled array

Below is a partially-filled array. When appending elements to this array in a loop, what statement(s) correctly updates the array with value?
 	
const size_t MAX = 100;
double nums[MAX];
size_t size = 0;
double value;
	nums[size++] = value;

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;
}
	If there is room to insert, the function returns false instead of true. It should say if (size == MAX)

What prints?
 	
int a[3][2] = {{3,2,3}};
cout << a[0][0] << a[1][0] << endl;
	30

Which value of a is stored in the val variable? auto val = a[0][2];	The value in the first row and the third column

What prints?
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;
	6

What prints?
 	
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;
	1

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;	

What prints?
 	
int x = 0;
int a[2][3] = {{1, 2, 3}, {4, 5, 6}};
for (auto r : a)
for (auto c : r) x++;
cout << x << endl;
6
	Illegal; will not compile

What is the value of a[1][2] after this runs?
 	
int cnt = 0, a[2][3];
for (int i = 0; i < 3; i++)
for (int j = 0; j < 2; j++)
a[j][i] = ++cnt;
	6

Which function prototype could process a 2D array?	void (f(int a[][2]);

Which statement displays the value 24 from the 2D array initialized here?
 	
int a[2][3] ={
{ 13, 23, 33 },
{ 14, 24, 34 }
};
	cout << a[1][1];

How many (int) elements are in this array? int a[2][3];	6

What prints?
 	
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;
	Illegal; will not compile

What happens here
 	
void f()
{
char * s = "CS 150";
s[0] = 'X';
cout << s << endl;
}
	Most likely crashes when run

What prints here?
 	
const char **a = "dog",** b = a;
if (strcmp(a, b)) cout << "dog == dog" << endl;
else cout << "dog != dog" << endl;
	NOT dog == dog

Which library function performs an equivalent operation on C-strings?
 	
string s1 = "Hello";
string s2 = "World";
s1 = s1 + s2;
	strcat()

What happens here?
 	
char s1[] = "CS150";
char *s2 = s1;
s2[0] = 'X';
cout << s1 << endl;
	"XS150"

Which library function performs an equivalent operation on C-strings?
 	
string s1 = f(), s2 = f();
if (s1 < s2) . . .
	strcmp()

Which library function performs an equivalent operation on C-strings?
 	
string s = mystery();
if (s.size() > 3) . . .
	strlen()

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";	

What happens here?
 	
char *s1 = "CS150";
char s2[] = s1;
s2[0] = 'X';
cout << s1 << endl;
	Does not compile

Where is the pointer s2 stored in memory?
 	
char s1[1024] = "Hello";
void f()
{
const char *s2 = "Goodbye";
char s3[] = "CS 150";
}
	stack

What is true about the command line in C++? 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[])	All of these are true
What is the correct version of main() if you wish to process command-line arguments?	int main(int argc, char* argv[])

The variable p points to:
 	
void f()
{
int *p = new int[3]{1, 2, 3};
}
	the first element of an array of 3 ints with the values 1,2,3

The variable *p:
 	
void f()
{
int *p = new int = {42};
}
 	
void f()
{
string *p = new string;
}
	1 is undefined. Code does not compile.
	
	2 stores an empty string

The variable p is located:
 	
void f()
{
int *p = new int;
}
 	
The variable p:void f(){ int *p = new int;}
	on the stack
	
	stores a memory address

Examine this code. What goes on the blank line?
 	
void f()
{
int *p = new int[3]{1, 2, 3};
. . . ______________
}
	delete[] p;

The variable *p:
 	
void f()
{
int *p = new int;
}
	is uninitialized

The variable *p:
void f()
{
int *p = new int{42};
}
	stores the value 42 in C++11 only

The variable p points to:
 	
void f()
{
int *p = new int[3] = {1, 2, 3};
}
	is undefined. Code does not compile.

void f()
{
int *p = new int{};
}	

Which line correct creates a smart pointer that points to the variable x? int x = 42;	None of these

What does this code print?
 	
int main()
{
auto p1 =unique_ptr<int>(new int{42});
cout << *p1;
auto p2 = p1;
cout << **p2; (**p2)++;
cout << *p2;
}
	Does not compile (illegal)

This code:
 	
void f()
{
int *p = new int[3]{rand(), rand(), rand()};
if (p[1] == 0 || p[2] == 0) throw "Divide by 0";
cout << p[0] / p[1] / p[2] << endl;
delete[] p;
}
	has a memory leak

What does this code print?
 	
int main()
{
auto p1 =unique_ptr<int>(new int{42});
cout << *p1;
auto p2 = p1.release();
cout << **p2; (**p2)++; cout << *p1;
}
	Undefined behavior

The member function get() returns the raw pointer that a smart pointer contains. What does this code print?
 	
int main()
{
auto p1 =unique_ptr<int>(new int{42});
cout << *p1;
auto p2 = p1.release();
cout << **p2; (**p2)++;
cout << p1.get() << endl;
}
	42420

This code:
 	
void f()
{
int *p = new int[3]{rand(), rand(), rand()};
if (p[1] == 0 || p[2] == 0) return;
cout << p[0] / p[1] / p[2] << endl;
delete[] p;
}
	has a memory leak

A user-defined type created as a struct	has its implementation as its interface

What is f()?
 	
class X
{
void f() const;
public:
X(int);
int g() const;
void h(int);
};
	NOT constructor

In C++ terminology, frequency() is called a:
 	
class Radio {
public:
Radio();
explicit Radio(double);
Radio(double, int);
 	
double frequency() const;
double frequency(double);
};
	accessor

class Time {
Time();
long get() const;
void set(long);
private: long seconds;
};	

Which element is private?
 	
struct Val
{
int data_;
public:
Val(int);
int get() const;
void print() const;
};
 	
void Val::get() { return data_; }
Val::Val(int n) { data_ = n; }
void Val::print() const { cout << data_; }
	None of these

Which element is private?
 	
class Val
{
int data_;
public:
Val(int);
int get() const;
void print() const;
};
 	
void Val::get() { return data_; }
Val::Val(int n) { data_ = n; }
void Val::print() const { cout << data_; }
	data_

Which of these is part of the implementation?
 	
class Alligator
{
public: Alligator(double w);
void eat();
string toString() const;
private:
double weight;
};
	weight

What is true about user-defined types implemented using classes with private data members?	modifications to the character of the data members does not require clients to rewrite code
What is true about an accessor member function?	It returns information about an object's state

On the second line of this code, the object named myRadio is:
 	
Radio myRadio(98.6, 8);
cout << myRadio.frequency() << endl;
	an implicit parameter

In C++ terminology, frequency(double) is called a:
 	
class Radio {
public:
Radio();
explicit Radio(double);
Radio(double, int);
double frequency() const;
double frequency(double);
 	
};
	mutator

In C++ terminology, the two members named frequency() are:
 	
class Radio {
public:
Radio();
explicit Radio(double);
Radio(double, int);
double frequency() const;
double frequency(double);
};
	member functions

Assume that you have the following code:
istreamstring in("one");int n;
Which of these (erroneous) statements cause the program to terminate?
	cout << stoi("one");
	assert(2 + 2 == 5);

A(n) ___ is an occurrence of an undesirable situation that can be detected during program execution.	exception

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

To count the number of elements in a vector that match a particular value, use the STL function:	count
Examine the following code (which is legal). What is the correct prototype for an aggregate output operator?struct Money { int dollars{0}, cents{0}; } m1, m2;	ostream& operator<<(ostream& out, const Money& m);

Given the following structure and variable definitions, which data members are default initialized?
 	
struct Employee
{
long empID;
std::string lastName;
double salary;
int age;
};
 	
Employee bob{777, "Zimmerman"};
	age
	salary

Which of the following lines is legal but undefined?
 	
enum class Coin
{ PENNY = 1, NICKEL = 5, DIME = 10, QUARTER = 25};
Coin c;
	c = static_cast<Coin>(.25);

Which array definition is illegal?
 	
const int SIZE = 3;
int a1[SIZE];int a2[3];
int a3[3]{};int a4[] = {1, 2, 3};
int a5[2] = {1, 2, 3};
	a5

What is true about this code?int * choice;	choice contains an undefined address

[1413] What does this loop do?
 	
int a[] = {6, 1, 9, 5, 1, 2, 3};
int x(0);
for (auto e : a)
x += e;
cout << x << endl;
	Sums the elements in a

What happens when this code fragment runs in C++ 11? cout << sqrt(-2) << endl;	sqrt() returns a not-a-number error value

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

What happens when you execute the following (erroneous) code: cout << stoi(42.5) << endl;	NOT The double 42.5 is truncated to 42 and printed
What happens when you execute the (erroneous) line: cout << stoi("fifteen") << endl;	NOT The conversion is impossible, so the code will not compile.

What happens when this code fragment runs?
 	
istringstream in("12.5");
int n;
in >> n;
	NOT It sets an error state in in.

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

What is x?
 	
vector<int> v{1, 2, 3};
auto x = min_element(v.begin(), v.end());
	an iterator

Which defines a vector to store the salaries of ten employees?	vector<double> salaries(10);

{
p = x 2;
return x / 2;
}
.
.
.
int a = 3, b, c;
c = f(&b, a);	

Which array definition is illegal (even if it may compile on some compilers)?
 	
int SIZE = 3;
int a1[SIZE];
int a2[3];
int a3[3]{};
int a4[] = {1, 2, 3};
int a5[3] = {1, 2};
	a1

What is stored in the last element of nums?int nums[3] = {1, 2};	0
Which expression returns the number of countries?string countries[] = {"Andorra", "Albania", . . . };	sizeof(countries) / sizeof(countries[0])
The _________ of an object is implemented by the object's member functions.	Behavior
____________ is the Object-Oriented design feature that allows you to create a new class by using an existing class as a starting point, extending it to add new attributes and behaviors.	Inheritance

What is h()?
 	
class X
{
public:
X(int);
void f() const;
int g() const;
void h(int);
};
	mutator

Which of these are part of the implementation?
 	
class Time
{
Time();
long get() const;
void set(long);
private:
long seconds;
};
	All of these are part of the implementation

What is X()?
 	
class X
{
public:
X(int);
void f() const;
int g() const;
void h(int);
};
	constructor

What is true about a mutator member function?	It changes one or more data members

In C++ terminology, frequency(double) is called a:
 	
class Radio
{
public: Radio();
explicit Radio(double);
Radio(double, int);
double frequency() const;
double frequency(double);
};
	mutator

What is f()?
 	
class X
{
void f() const;
public:
X(int);
int g() const;
void h(int);
};
	None of these

class Alligator
{
public:
Alligator(double w);
void eat();
string toString() const;
private: double weight;
};	

Which of these is a constructor?
 	
class Alligator
{
public:
Alligator(double w);
void eat();
string toString() const;
private:
double weight;
};
	Alligator()

The following code:
 	
#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");
}
	Compiles but does not link.

There is no constructor for this class.
 	
class Integer
{
int value_;
public: int get() const;
int set(int n);
};
	You can create objects; value_ is uninitialized

What prints here?
 	
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;
}
	Does not compile; c1 is not an object

A Fraction denominator must not ever become 0. You can enforce this invariant through:
 	
class Fraction
{
. . .
public:
Fraction(int, int);
Fraction get() const;
Fraction set(int, int);
};
	the implementation of the mutator member

class Alligator
{
public:
Alligator(double w);
void eat();
string toString() const;
private:
double weight;
};	

Which of these is an accessor?
 	
class Alligator
{
public:
Alligator(double w);
void eat(); string toString() const;
private: double weight;
};
	toString()

What is g()?class X{public: X(int); void f() const; int g() const; void h(int);};	accessor
Which of these is not a property of an object (in the OOP sense)?	Substitutablility
What happens with this code?#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_; }int main(){ string s = "Bill"; Cat b; b = s; cout << b.get() << endl;}	NOT Line b = s; does not compile. Type mismatch

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?
 	
BigNum a{"12345.795"}, b{".95873421"};
auto c = a * b;
	const BigNum operator*(const BigNum& lhs, const BigNum& rhs);

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.) 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 ____________________; }};	Does not compile, changes arity of operator.
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.) 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.

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?
 	
BigNum a{"12345.795"}, b{".95873421"};
const BigNum operator-(const BigNum& n) {...}
	auto c = -b;

What happens with this code?
#include <string>
#include <iostream>
using namespace std;
 	
class Cat
{
string name_;
public:
Cat() = default;
Cat(const string& n);
string get() const;
};
 	
Cat::Cat(const string& n): name_(n) {}
string Cat::get() const { return name_; }
 	
int main(){ string s = "Bill";
Cat b; b = s; cout << b.get() << endl;
}
	NOT Line Cat b; does not link. No suitable implementation. Maybe The does compile; it prints "Bill".

What statement about constructors is false?	You must write at least one constructor for every class

ForX::opeartor^= ForX::operator& operator& side-effect operator	ternary member operator unary non-member operator (correct) ForX::operator^=

Which operator is implemented here? (No class prototypes shown.)
 	
clas ForX{
itn data;
};
//////////////???///////////////////////
{
data -= rhs.data;
return *this;
}
 	
No prototypes are shown inside the class. Assume that a and b are objects of type ForX.
	a -= b

1.The Date class represents a day on a calendar. Examine the code shown. Which operator is called?Date d{2018, 7, 4};auto e = ++d;
 	
2.The Date class represents a day on a calendar. Examine the code shown. Which operator is called?Date d{2018, 7, 4};auto e = d++;
 	
3. The Date class represents a day on a calendar. Examine the code shown. Which operator is called?Date d{2018, 7, 4};auto e = d++;
	Date& operator++(Date&); NOT Date& Date::operator++(int);
	
	const Date Date::operator++(int);
	
	const Date operator++(Date&, int);

Which operator is implemented here? (No class prototypes shown.)
 	
clas ForX{
itn data;
};
//////////////???///////////////////////
{
--data;
return *this;
}
 	
No prototypes are shown inside the class. Assume that a and b are objects of type ForX.
	--a

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) { . . . }};const Point operator+(Point& lhs, const Point& rhs) { return lhs += rhs;}	The operator should not change any of its parameters
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 operator return type should be a Point&
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) { . . . }};Point& operator+(const Point& lhs, const Point& rhs) { return Point(lhs) += rhs;}	The operator return type should not be a Point&.
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.) 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 _________________;}	Does not compile; cannot change data members of object; no mutators.
The operators that cannot be overloaded are _____, _____, _____, and _____.	(.), (?:), (.*), and (::)

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

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;}
	Compiles, links: prints 5510

#include <string>
#include <iostream>
using namespace std;
class Cat
{
string name_;
public:
Cat() = default;
Cat(const string& n);
string get() const;
};
 	
Cat::Cat(const string& n): name_(n) {}
string Cat::get() const { return name_; }
 	
int main()
{
string s = "Bill";
Cat b; b = s;
cout << b.get() << endl;
}	

1. The copy constructor for this class is: class Radio {public: Radio(); explicit Radio(double); Radio(double, int); double frequency() const; double frequency(double);}; 2.The conversion constructor for this class is: class Radio {public: Radio(); explicit Radio(double); Radio(double, int); double frequency() const; double frequency(double);};	1.None of these 2.Radio(double)
The Point class represents x,y coordinates in a Cartesian plane. Which line of code appears completes this operator which returns the address of the Point object in memory? (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& p) { return ___________;}	Does not compile; uses a non-operator symbol.
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() << ')';}	There is no error; it works fine.

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?
BigNum a{9.2573e27};auto c = a / 100.0;
 	
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?
BigNum a{9.2573e27};auto c = 100.0 / a;
 	
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?
BigNum a{"12345.795"}, b{".95873421"};
const BigNum BigNum::operator-(const BigNum& n) const {...}
	All of these can be used
	
	const BigNum BigNum::operator/(const BigNum& rhs) const;
	
	auto c = a - b

What happens with this code?#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_; }int main(){ string s = "Bill"; Cat b; b = s; cout << b.get() << endl;}	Line Cat b; does not compile. No suitable constructor.
The fstream class is the/a _______ class of istream. The ostream class is the/a _______ class of ofstream. The ostream class is the/a _______ class of ios. The ostream class is the/a _______ class of istream.	descendent base derived sibling
What does a derived class inherit from its base class?	Both data and behavior
Which among the following is the legal way of implementing the constructor of the Manager class that passes parameters to a base-class constructor? class Employee {public: Employee(); Employee(const string&); Employee(double); Employee(const string&, double);private: string name; double salary;};class Manager : public Employee {public: Manager(); Manager(const string& d, const string& n, double s);private: string department;};	Manager::Manager(const string& d, const string& n, double s): Employee(n, s) { department = d; }

```
class Car
{
public: Car() = default;
Car(double s): speed(s) {}
double getSpeed() const { return speed; }
private: double speed = 0;
};

class AeroCar : public Car
{
public:
AeroCar() = default;
AeroCar(double h, double s) : Car(s * 2), height(h) {} void display() const;

private: double height = 0;
};

void AeroCar::display() const
{
cout << "Speed: " << getSpeed() << ";
Height: " << height << endl;
}

int main(){ AeroCar acar1(2000, 200); acar1.display();}
```

Which of these is an example of the principle of substitutability?

void f1(fstream& out) { . . .}void f2(int n) { . . . }void f3(const string& s) { . . . }void f4(ios& i) { . . . }

f4(cout);

Suppose that we have a function that registers a Vehicle object. We also have a Car object that is a specialized Vehicle (defined by inheritance). The substitution principle states _______.

The Car object can be used in the Vehicle registration function because it is a kind of Vehicle.

Based on the following declaration of the Employee class where Manager is derived from Employee, which of the following are true?

```
class Employee
{
public: Employee();
Employee(const string&);
Employee(double);
Employee(const string&, double);
void setName(const string&);
string getName()const;
private: string name; double salary;
};
```

The Manager class inherits name and salary, but Manager functions can only change the values of the name data member.

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

Assume you have a Student object named bill. Which of these statements would be legal?

```
bill.name = "Bill Gates"; // I
bill.setName("Bill Gates"); // II
cout << bill.getName(); // III
bill.studentID = 123L; // IV
cout << bill.getID(); // V
```

II, III, V

Which one of the following is an example of the "substitution principle"?

A derived class object can be used in place of a base-class object.

ChoiceQuestion is derived from the Question base class . ChoiceQuestion overrides the display() function defined in the Question base class. Which of the following will call the base class display() function from the ChoiceQuestion display() function?

Question::display()

Examine the following UML diagram.

Which of these data members or member functions are inherited (and accessible) by the Student class??

getName(), setName()

---

score() pure virtual function
class Hand abstract base class
Hand() default constructor
sort()

...

Which statement displays the value 24 from the 2D array initialized here?

```
int a[2][3] ={ { 13, 23, 33 }, { 14, 24, 34 }};
```

cout << a[1][1];

What is printed?

```
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;}

int main(){ int a[] = {1, 2, 5, 2, 5, 4}; cout << mystery(a, 6) << endl;}
```

None of these

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;}	

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?
 	
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;
}
	The condition (sizeA > sizeB) should be (sizeB > sizeA)

What is true about this inheritance hierarchy (using both C++ and traditional, classic OO terminology)? Assume that Party has a member function (which Person overrides), declared as: virtual void print() = 0;	Party is a base class Party is an abstract class Organization is a subclass Person is a derived class Organization is a superclass
Examine the class hierarchy below. Assume that both derived classes are concrete and completely defined. Which of the following member functions are the derived classes allowed (but not required to) override? class Hand { std::vector<Card> cards;public: Hand() = default; virtual ~Hand() = default; void add(const Card&); virtual int score() const = 0; virtual void sort(); bool operator<(const Card& rhs) const;};class PokerHand : public Hand { . . . }class BlackjackHand : public Hand { . . . }	sort()
Given this declaration, which line below is illegal? auto p1 = unique_ptr<int>(new int{42});	auto p2 = p1;
The following code:#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;}	Does not compile.
Below is a class hierarchy for card games. Assuming that these are the only classes and that the concrete classes are correctly completed, which of the following definitions will not compile? class Hand { std::vector<Card> cards;public: void add(const Card&); Card get(size_t index) const; virtual int score() const;};class PokerHand : public Hand { . . . };class BlackjackHand : public Hand { . . . };class GoFishHand : public Hand { . . . };	PokerHand* = new Hand;
Below is a class hierarchy. Which assignments are illegal? class Widget { . . . };class Label: public Widget { . . . };class Button: public Widget { . . . };class Text: public Widget { . . . };class TextArea: public Text { . . . };class TextLine: public Text { . . . };class Container: public Widget { . . . };class Canvas: public Container { . . . };class Window: public Container { . . . };	Canvas* p = new Container;
_____________ is one of the primary mechanisms that we use to understand the natural world around us. Starting as infants we begin to recognize the difference between categories like food, toys, pets, and people. As we mature, we learn to divide these general categories or classes into subcategories like siblings and parents, vegetables and dessert.	classification
What is the primary purpose of inheritance?	Model similar objects with different behavior.

```
#include <iostream>
using namespace std;
class Shape
{
public:
string toString() const { return "Shape"; }};

class Circle : public Shape
{
public: string toString() const { return "Circle"; }
};

class Triangle : public Shape
{
public: string toString() const { return "Triangle"; }
};

int main()
{
Shape* s1 = new Circle;
Shape* s2 = new Triangle;
cout << s1->toString() << s2->toString() << endl;
}
```

2 What prints when this code is run?#include <string>#include <iostream>using namespace std;

```
class Shape
{
public: virtual void iam() const; };

class Square : public Shape { };

class Oval: public Shape { public: void iam() const; };

void Shape::iam() const { cout << "Shape"; }
void Oval::iam() const { cout << "Oval"; }
void iam(const Shape* s) { s->iam(); }
```

int main() { Shape **a = new Shape,** b = new Square, *c = new Oval; iam(a); iam(b); iam(c);}

2 ShapeShapeOval

The Time class represents the time of day on a clock. Examine the code shown. Which operator is called?Time t(8, 30, "a");cout << t << endl;

ostream& operator<<(ostream&, const Time&);

This code:

```
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;
}
```

None of these

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?

```
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;
}
```

size should not be passed by reference

What prints?

```
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;
```

14

What does this function do?

```
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;
}
```

Returns the index of the last occurrence of the smallest number in the array

CS150 Midterm 3

Which loop is used when deleting an element from an array?

for (j = pos; j < size; j++) a[j] = a[j + 1];

To use any of C++ smart pointer types, include the header:

<memory>

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;
}
```

has a dangling pointer

The _______ of a class specifies how clients interact with a class.

public interface

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.)

```
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 ___________________; }
};
```

All of these will work

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

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

To create the GiPod() from the Cadillac class, you should use.

NOT public inheritance

Below is a class hierarchy for card games. Assuming that these are the only classes and that the concrete classes are correctly completed, which of the following non-member functions are polymorphic?

```
class Hand
{
std::vector<Card> cards;
public: void add(const Card&);
Card get(size_t index) const;
virtual int score() const;
};
class PokerHand : public Hand { . . . };
class BlackjackHand : public Hand { . . . };
class GoFishHand : public Hand { . . . };
```

void draw(const Hand& h) { . . . }

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;
```

sum->8

What does this code mean?class X : public Y { . . .};

Every X object is-a Y object

non-member draw() function, assuming that the function makes use of the virtual functions overridden in PokerHand?
 	
class Hand
{
std::vector<Card> cards;
public: Hand() = default;
virtual ~Hand() = default;
void add(const Card&);
virtual int score() const = 0;
virtual void sort();
bool operator<(const Card& rhs) const;
};
 	
class PokerHand : public Hand { . . . }
class BlackjackHand : public Hand { . . . }
void draw(const Hand h) { . . . }	

Below is a class hierarchy for card games. Assuming that these are the only classes and that the concrete classes are correctly completed, which of the following definitions will not compile?
 	
class Hand
{
std::vector<Card> cards;
public:
void add(const Card&);
Card get(size_t index) const;
virtual int score() const;
};
class PokerHand :public Hand{ . . . };
class BlackjackHand : public Hand { . . . };
class GoFishHand : public Hand { . . . };
	BlackjackHand* h = new Hand;

Which member function(s) in Hobbit cause a compiler error?
 	
class Creature {
public:Creature(const string& name);
virtual string name() const final;
virtual string skills() const;
virtual void addSkill(const string& skill);
void print() const;};
class Hobbit : public Creature {public:string name() const override;
string skills() const override;
void addSkill(const string&) override;
void print() override;};
	name(), print()

Which member function(s) must be overridden in Hobbit?
 	
class Creature {public:Creature(const string& name);
virtual string name() const final;
virtual string skills() const;
virtual void addSkill(const string& skill);
void print() const;};
class Hobbit : public Creature {. . .};
	None of them

What prints when this code is run?
#include <string>
#include <iostream>
using namespace std;
 	
class Shape { public: virtual void iam() const; };
 	
class Square : public Shape { public: void iam() const; };
 	
class Oval: public Shape { public: void iamm() const; };
 	
void Shape::iam() const { cout << "Shape"; }
 	
void Square::iam() const { cout << "Square"; }
 	
void Oval::iamm() const { cout << "Oval"; }
 	
void iam(const Shape* s) { s->iam(); }
 	
int main()
{
Shape a = new Shape, b = new Square, *c = new Oval;
 	
iam(a);iam(b);iam(c);
 	
}
	ShapeSquareShape