 **This quiz has been regraded; your new score reflects 2 questions that were affected.**

**MIDTERM EXAM**

* **Due** Mar 17 at 2:40pm

* **Points** 100

* **Questions** 50
* **Available** Mar 17 at 12:40pm - Mar 17 at 2:40pm about 2 hours

* **Time Limit** 120 Minutes
* **Allowed Attempts** 2

This quiz was locked Mar 17 at 2:40pm.

Attempt History

|  | **Attempt** | **Time** | **Score** | **Regraded** |
| --- | --- | --- | --- | --- |
| **LATEST** | [Attempt 1](https://dvc.instructure.com/courses/27283/quizzes/47693/history?version=2) | 117 minutes | 80 out of 100 | 82 out of 100 |

 Correct answers are hidden.

Score for this attempt: **82** out of 100

Submitted Mar 16 at 9pm

This attempt took 117 minutes.

**Question 1**

**2 / 2 pts**

C++ class is similar to one of these.



library function



inline function



structure



header file

**Question 2**

**2 / 2 pts**

Examples of access specifiers are the keywords:



table and raw



private and public



opened and closed



near and far

**Question 3**

**2 / 2 pts**

Objects are created from abstract data types that encapsulate \_\_\_\_\_\_\_ and \_\_\_\_\_\_\_ together.



integers and floats



addresses, pointers



numbers, characters



data and functions

**Question 4**

**2 / 2 pts**

Class declarations are usually stored here.



In .cpp files, along with function definitions



Under pseudonyms



In their own header files



On separate disk volumes

**Question 5**

**2 / 2 pts**

When the body of a member function is defined inside a class declaration, it is said to be



globally



inline



static



conditionally

**Question 6**

**2 / 2 pts**

\_\_\_\_\_\_\_\_\_\_\_ is a member function that is automatically called when a class object is \_\_\_\_\_\_\_\_\_\_\_.



utility function, declared



static function, de-allocated



constructor, created



destructor, created

**Question 7**

**2 / 2 pts**

The constructor function's return type is



char



int



None of the given answers



structure pointer



float

**Question 8**

**2 / 2 pts**

The destructor function's return type is:



tilda



int



float



Nothing, destructors have no return type

**Question 9**

**2 / 2 pts**

When a constructor function accepts no arguments, or does not have to accept arguments because of default arguments, it is called a(n):



arbitrator function



empty constructor



stand-alone function



default constructor

**Question 10**

**2 / 2 pts**

This type of member function may be called from a statement outside the class.



private



undeclared



public



framed

**IncorrectQuestion 11**

**0 / 2 pts**

What is the answer of the following postfix expression (Hint:  use a stack to find the answer)?

10 2 8 \* + 3 -



29



23



17



20

**IncorrectQuestion 12**

**0 / 2 pts**

A variable named this is automatically available in every member function of a class and does not need to be declared explicitly. Inside a member function for a class X, what is the type of the variable this?



X\*



void\*



X



X&

**IncorrectQuestion 13**

**Original Score: 0 / 2 pts Regraded Score: 0 / 2 pts**

**This question has been regraded.**

|  |
| --- |
| Suppose we have a class Thing declared in file thing.h as follows.    class Thing {  public:  // initialize this Thing when it is constructed  Thing();  // other public operations  ...  private:  // private representation  ...  };    Now suppose we execute the following main program    #include "thing.h"    int main() {  Thing it;  Thing them[2];    // quit without doing anything else  return 0;  }    How many times is the default (null) constructor for class Thing executed? |



1



2



3



4

**Question 14**

**2 / 2 pts**

Consider the following class definition, which represents an Employee, and the implementation of one of the methods:

class Employee {  
private:  
string name;  
…  
public:  
…  
bool equals(Employee& other);  
};

bool Employee::equals(Employee& other) { return name == other.name; }

Is there anything wrong with the implementation of Employee::equals?



This method is useless since one Employee object could never equal another.



There is no problem



ThThe use of other.name is not valid because it does not belong to the local Employee object associated with the function call.e equals method as implemented does not return value of type bool. It returns a –1, 0, or 1.



The use of other.name is not valid because it does not belong to the local Employee object associated with the function call.

**Question 15**

**2 / 2 pts**

Consider the following class definition which represents a geometric Point:

class Point {

private:

// representation is here

public:

Point(double x, double y);

double getX();

double getY();

double distanceTo(Point other); // distance between this point and the other

void paint(); // paint the Point on the screen

};

From the definition of Point above we know that the representation of a Point must be:



Two variables of type double: one for the x coordinate, and one for the y coordinate.



A dynamically allocated array.



A polar representation: an angle and a distance from the origin.



There is no way of knowing.

**Question 16**

**2 / 2 pts**

**Given the following:**

**void cse143(double c){ cout << "Fe" << endl;}**

**void cse143(string c){ cout << "Fi" << endl;}**

**void cse143(int c) { cout << "Fo" << endl;}**

**void cse143(int \*c) { cout << "Fum" << endl;}**

**What output does the following function call produce?**

**cse143(17);**



Fo



Fe



Fi



Fum

**IncorrectQuestion 17**

**0 / 2 pts**

**Suppose we have the following class containing a constructor, copy constructor, assignment operator, and destructor:**

**class X {**

**public:**

**X(); // constructor**

**X(X &value); // copy constructor**

**X & operator=(X &value); // assignment**

**~X(); // destructor**

**private:**

***hidden representation***

**};**

Now, consider the following small program. What sequence of function calls occurs when this program is executed?

int main( ) {

X a;

X b = a;

a = b;

return 0;

}



X(), X(), operator=, copy constructor, ~X(),~X()



X(), copy constructor, operator=, ~X(),~X()



X(), X(), operator=, operator=, ~X(),~X()



X(), operator=, copy constructor, ~X(),~X()

**Question 18**

**2 / 2 pts**

Suppose the following class implements a stack of integers. Also, suppose that we have a private members that keep track of the top of the stack and the bottom of the stack.

class IntStack

{

public:

IntStack( );

bool isEmpty( ); // returns true if the stack is empty

void push(int item); // adds an element to the stack

int pop( ); /// deletes an element from the stack

/// The top function returns the value of the top of the stack but does ///not delete the top element

int top( );

}

What happens if we execute the following statements?

IntStack s;

int n1, n2, n3;

s.push(17);

s.push(143);

s.push(42);

n1 = s.pop( );

n2 = s.top( );

s.push(n1);

n3 = s.pop( );

n1 = s.top( );



Stack is empty; n1=17, n2=143, n3=42



Stack contains 143 (top), 17 (bottom); n1=143, n2=143, n3=42



Stack contains 42 (top), 17 (bottom); n1=42, n2=143, n3=143 13.



Stack contains 143 (top), 17 (bottom); n1=42, n2=42, n3=42

**Question 19**

**2 / 2 pts**

In the following code, assume the myStack object is a stack that can hold integers, and that value is an int variable. (The lines are numbered for reference purposes.)

1 myStack.push(0);

2 myStack.push(1);

3 myStack.push(2);

4 myStack.pop(value);

5 myStack.push(3);

6 myStack.pop(value);

7 cout << value << endl;

Assume that the pop function, called in lines 4 and 6, stores the number popped from the stack in the value variable.

What will the statement in line 7 display?



3



0



1



2

**Question 20**

**2 / 2 pts**

In the following code, assume the myQueue object is a queue that can hold integers, and that value is an int variable. (The lines are numbered for reference purposes.)

1 myQueue.enqueue(0);

2 myQueue.enqueue(1);

3 myQueue.enqueue(2);

4 myQueue.dequeue(value);

5 myQueue.enqueue(3);

6 myQueue.dequeue(value);

7 cout << value << endl;

Assume that the dequeue function, called in lines 4, and 6, stores the number removed from the queue in the value variable.

What will the statement in line 7 display?



0



1



3



2

**Question 21**

**Original Score: 0 / 2 pts Regraded Score: 2 / 2 pts**

**This question has been regraded.**

A linked list class must take care of removing the dynamically allocated nodes. This is done by \_\_\_\_\_\_\_\_



a. overriding the removal function



b. overloading the memory persistence operator



c. the destructor function



d. the constructor function

**Question 22**

**2 / 2 pts**

* **One of your classmates is trying to debug a list class that uses an array to hold the**

**list elements. Something is wrong somewhere in this code. Where are the bug(s)?**

**Chose the correct sequence of lines where the bugs are .**

* 1. **const int MAXSIZE = 1234; // max # elements that can be stored in a list**
  2. **class List {**
  3. **public:**
  4. **// construct empty list**
  5. **List( ) { size = 0; }**
  6. **// insert item at end of list; do nothing if list is already full**
  7. **void Insert(string item) {**
  8. **if (size < MAXSIZE) {**
  9. **size++;**
  10. **data[size] = item;**
  11. **}**
  12. **}**
  13. **// = # of elements in this list**
  14. **int sizeOf( ) {**
  15. **return size;**
  16. **}**
  17. **// = location of item in the list, or –1 if not found**
  18. **int find(string item) {**
  19. **int k = 0;**
  20. **while (data[k] != item && k < size)**
  21. **k++;**
  22. **if (k < size) return k; else return –1;**
  23. **}**
  25. private:
  26. **int size; // # of elements currently stored in this list**
  27. **string data[MAXSIZE];**
  28. **};**



20, 21, 22, 23



9, 10, 20



4, 5, 6



9, 10, 11, 12

**Question 23**

**2 / 2 pts**

What is the value of the postfix expression 6 3 2 4 + - \*



Something between 5 and 15



Something between 5 and -5



Something between -15 and -100



Something between -5 and -15



Something between 15 and 100

**Question 24**

**2 / 2 pts**

Which of the following statement(s) about stack data structure is/are NOT correct?



Top of the Stack always contain the new node



Stack is the FIFO data structure



Null link is present in the last node at the bottom of the stack



Linked List are used for implementing Stacks

**Question 25**

**2 / 2 pts**

What kind of linked list is best to answer question like “What is the item at position n



Doubly Linked List



Array Implementation of a Linked List



Circular Linked List



Singly Linked List

**Question 26**

**2 / 2 pts**

Linked list is considered as an example of \_\_\_\_\_\_\_\_\_\_\_ type of memory allocation.



None of the mentioned



Dynamic



Static



Compile Time

**Question 27**

**2 / 2 pts**

Consider the following pseudocode:

declare a stack of characters

while ( there are more characters in the word to read )

{

read a character

push the character on the stack

}

while ( the stack is not empty )

{

write the stack's top character to the screen

pop a character off the stack

}

What is written to the screen for the input "carpets"?



carpets



serc



ccaarrppeettss



steprac

**Question 28**

**2 / 2 pts**

In the following statement, what does T represents?

template < class T >



The int data type



"T" stands for "Template"



The name of the function template



A generic data type that is used in a function template

**Question 29**

**2 / 2 pts**

Here is an INCORRECT pseudocode for the algorithm which is supposed to determine whether a sequence of parentheses is balanced:

declare a character stack

while ( more input is available)

{

read a character

if ( the character is a '(' )

push it on the stack

else if ( the character is a ')' and the stack is not empty )

pop a character off the stack

else

print "unbalanced" and exit

}

print "balanced"

Which of these unbalanced sequences does the above code think is balanced?



( ( ( ) )



( ( ) ( ) ) )



( ( ) ) ) ( )



( ) ) ( ( )

**Question 30**

**2 / 2 pts**

If you do not declare an access specification, the default for members of a class is



inline



private



gloval



public

**Question 31**

**2 / 2 pts**

Assuming that Rectangle is a class name, the statement

Rectangle \*BoxPtr;



Assigns the value of \*BoxPtr to the object Rectangle



defines a Rectangle pointer variable called BoxPtr



is illegal in C++



Declares an object of class Rectangle

**IncorrectQuestion 32**

**0 / 2 pts**

When you dereference an object pointer, use the



-> operator



& operator



dot operator



<> operator

**Question 33**

**2 / 2 pts**

A class may have this many default constructor(s).



A maximum of two



Any number of



Only one



More than one

**Question 34**

**2 / 2 pts**

When a member function is defined outside of the class declaration, the function name must be qualified with the:



Class name followed by the scope resolution operator



private access specifier



Class name followed by a semicolon



name of the first object

**Question 35**

**2 / 2 pts**

For the following code, which statement is *not* true?

class Point

{

private:

double y;

double z;

public:

double x;

};



x, y, and z are called members of the class



x is available to code that is written outside the class.



z is available to code that is written outside the class



The name of the class is Point

**Question 36**

**2 / 2 pts**

Assume that myCar is an instance of the Car class, and that the Car class has a member function named accelerate. Which of the following is a valid call to theaccelerate member function?



myCar.accelerate();



myCar:accelerate();



Car->accelerate();



myCar::accelerate();

**Question 37**

**2 / 2 pts**

What is the output of the following program?

#include <iostream>

using namespace std;

class TestClass

{

public:

TestClass(int x)

{ cout << x << endl; }

TestClass()

{ cout << "Hello!" << endl; }

};

int main()

{

TestClass test;

return 0;

}



This is not a C++ code



The program runs, but with no output



Hello!



0

**Question 38**

**2 / 2 pts**

What is the output of the following program?

#include <iostream>

using namespace std;

class TestClass

{

public:

TestClass(int x)

{ cout << x << endl; }

TestClass()

{ cout << "Hello!" << endl; }

};

int main()

{

TestClass test(77);

return 0;

}



This is not a C++ code



77



Hello!



The program runs but no output

**Question 39**

**2 / 2 pts**

What is the output of the following program?

#include <iostream>

using namespace std;

class TestClass

{

private:

int val;

void showVal()

{ cout << val << endl; }

public:

TestClass(int x)

{ val = x; }

};

int main()

{

TestClass test(77);

test.showVal();

return 0;

}



0



The program will not compile



The program runs, but with no output



77

**Question 40**

**2 / 2 pts**

struct entry

{

int num;

entry \* next;

};

entry \* head, \* cur, \* pent;

int n = 5, cnt=0;

pent = new entry;

pent->num = 7;

pent->next = NULL;

**for (cur=head; cur!=NULL; cur = cur->next)**

**{**

**if (cur->num == n)**

**cout << “hi” << endl;**

**}**

If entry, cur, head and n are defined as earlier and head points to a linked list of numbers, the above code displays “hi” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



For each entry in the linked list



Only for the last occurrence of the variable n in the linked list



Only for the first occurrence of the variable n in the linked list.



For all occurrences of the variable n in the linked list

**Question 41**

**2 / 2 pts**

The advantage a linked list has over a vector is:



Data removal and insertion are more accurate with a linked list than with a vector



A linked list can dynamically shrink or grow, and a vector cannot



A node can be inserted into or removed from a linked list faster than from a vector



A linked list is smaller than a vector

**IncorrectQuestion 42**

**0 / 2 pts**

Which keyword can be used in template?



class



class and typename



function



class and function



typename

**Question 43**

**2 / 2 pts**

Consider the following code  
  
template <class T> class className { ... .. ... public: T var; T someOperation(T arg); ... .. ... };  
  
Which of the following is NOT a correct way to create an object



className<T> classObject;



className<int> classObject;



className<float> classObject;



className<string> classObject;

**Question 44**

**2 / 2 pts**

A "tail" pointer in a queue implementation is:



Depends on the compiler



A must



Useful for better performance



Unnececary

**IncorrectQuestion 45**

**0 / 2 pts**

A "tail" pointer in a linked stack implementation is:



Unnecessary



Depends on the compiler



Useful for better performance



Esential

**IncorrectQuestion 46**

**0 / 2 pts**

What do "assertions" do?



Outputs a warning during runtime if the contents in its parentheses does not match



Terminate the program if the content in its parentheses is true



Outputs a warning during compilation if the contents in its parentheses does not match



Terminate the program if the content in its parentheses is false

**IncorrectQuestion 47**

**0 / 2 pts**

**struct entry**

**{**

**int num;**

**entry \* next;**

**};**

**entry \* head, \* cur, \* pent;**

**int n = 5, cnt=0;**

**pent = new entry;**

**pent->num = 7;**

**pent->next = NULL;**

**for (cur=head;cur!=NULL;cur=cur->next)**

**{**

**if (cur->num >= n)**

**break;**

**cnt++;**

**}**

**cout << cnt<<endl;**

**If entry, cur, head, num and cnt are defined as earlier and head points to a linked list of numbers, the above code displays**



The total number of entries which has a number greater than n



The total number of entries which has a number less than n.



The total number of entries which has a number less than n if head is pointing to an ordered linked list



The total number of entries which has a number greater than n if head is pointing to an ordered linked list.



The total number of entries in the linked list if head is pointing to an ordered linked list

**Question 48**

**2 / 2 pts**

**struct entry**

**{**

**int num;**

**entry \* next;**

**};**

**entry \* head, \* cur, \* pent;**

**int n = 5, cnt=0;**

**pent = new entry;**

**pent->num = 7;**

**pent->next = NULL;**

**pent->next=head;**

**head=pent;**

**If cur,head and pent are defined as earlier code above and head points to a linked list, the code fragment in this problem,**



adds a node in increasing order



removes an entry from the head



adds an enty when only one node is present in the list



adds an entry at the head

**Question 49**

**2 / 2 pts**

"Data hiding" is when:



A main program has access to public data members of an object, but is not supposed to use them directly.



Data members in a class are used, but are not declared anywhere.



A main program does not have direct access to an object's data members and has to use getters and setters instead.



A main program has no way at all to get or set hidden data in an object.

**Question 50**

**2 / 2 pts**

Array, Linked List, Stacks and Queues are linear data structures



True



False

Quiz Score: **82** out of 100