**1. Answer the questions below**

(a) the differences between a struct and a class are:

* Classes are more secure than struct, and the members of class are private by default and struct are public by default. (“Structure vs Class in C++ - GeeksforGeeks”)
* Classes are stored in the heap while struct are stored in the stack.
* Classes are reference type while struct are value type.
* Classes provides re-usability by using inheritance and provides flexible

combining data and methods.

* Default access to specifier is private in a class while struct default access to

the specifier is public.

**(b) What is the difference between a class and an object?**

Class is a blueprint (template) for creating an object and an object is an instance of a class.

**(c) What is the difference between inheritance and composition? Which one should you**

**lean towards when designing your solution to a problem?**

* Inheritance is creating a new class (derived class) from an existing one (base class)
* Composition defines a class as the sum of its parts OR used features of an existing class inside your new class OR using smaller objects to create a bigger one.

**(d) What is the difference between a deep vs a shallow copy? What can you do to make**

**one or the other happen?**

* Deep Copy:

An object is created by copying data of all variables and it also allocates similar memory resources with the same value to the object. (“Shallow Copy and Deep Copy in C++ - GeeksforGeeks”)

OR

It’s the copy of the object itself. A new memory is allocated for the object and contents are copied.

* Shallow Copy

An object created by simply copying the data of all variables of the original object.

OR

It’s copying the reference pointer of the object which means the new object is pointing to the same memory reference of the old object.

**(e) What is the difference between a constructor and a destructor ? Are they both manda-**

**tory or even necessary?**

* A class constructor is a member function of a class that is executed whenever we create an object of that class and it helps to initialize the object of a class.
* Destructor is a member function like constructor and has the same name as the class preceded by a tilde ~ but it helps to deallocate the memory of an object.

(“C++ Class Constructor and Destructor - Tutorialspoint”)

**(f) What is static vs dynamic typing? Which does C++ employ and which does Pythona employ?**

* C++ is a good example of a static typed language in which you have to declare the variables data type before you run your program.
* Python is a good example as well for dynamic language and it’s the opposite of static typing, which in Python you have to define the variables before you use it. (“Static vs Dynamic Typing | PERPETUAL ENIGMA”)

**(g) What is encapsulation vs abstraction? Please give some examples!**

Encapsulation is bundling all data and the functions together. or wrapping data and functions unders a single unit.

Abstraction is hiding the details from the user and only providing the essential information.

**(h) What is the difference between an abstract class and an interface?**

Abstract classes is declared as abstract and we declare the class as abstract and we don’t implement anything, but the class that inherits from this class have to provide a definition for the pure functions.

Interfaces is a class that has no implementation and only has a destructor and pure functions. (“Pure Virtual Functions and Abstract Classes in C++ - GeeksforGeeks”)

**(i) What is the difference between a virtual function and a pure virtual function?**

A virtual function is a member function which is declared within base class and is re-defined by a derived class.

A pure virtual function in C++ is a virtual function for which we don’t have an implementation, we only declared it. (“Pure Virtual Functions and Abstract Classes in C++ - GeeksforGeeks”)

**(j) What is the difference between Function Overloading and Function Overriding?**

Function overloading is when multiple functions hsa the same name in the same class.

Function overriding is override a function from a parent class in the child class.

**2. Deﬁne the following and give examples of each :**

1. **Polymorphism** is the ability of different functions to be invoked with the same name. There are two forms. (Griffin)
2. **Encapsulation** is grouping data and functions together and keeping their implementation details. (Griffin)
3. **Abstraction** is hiding the details from the user and only providing the essential information. (Griffin)

**3. Answer the Questions:**

1. **What is a default constructor?**

Default constructor: is a constructor that can be called with no arguments and empty parameters list. (“Default Constructors - Cppreference.Com”)

1. **What is an overloaded constructor? And is there a limit to the number of overloaded constructors you can have?**

Overloading constructor essentially have the same name (name of the class) and different number of arguments. No there is no limit for overloading constructor because sometimes the user needs to enter one info at the time.

1. **What is a copy constructor? Do you need to create a copy constructor for every class you deﬁne?**

Copy Constructor is a copy constructor is a member function that initializes an object using another object of the same class. No you don’t need to make a copy constructor every class you define because the compiler creates a default copy constructor for each class. (“Copy Constructor in C++ - GeeksforGeeks”)

1. **What is a deep copy, and when do you need to worry about it?**

Deep Copy: It’s the copy of the object itself. A new memory is allocated for the object and contents are copied.

1. **Is there a relationship between copy constructors and deep copying?**

Deep copy needs a Copy Constructor because it helps you allocate memory for the new deep copy.

1. **Is a copy constructor the same as overloading the assignment operator?**

Copy constructor and the assignment operator are almost equivalen, both copy one object to another. (“13.15 — Overloading the Assignment Operator | Learn C++”)

1. **Give one or more reason(s) why a class would need a destructor.**

A class needs a destructor to help deallocate memory and do other cleanup for a class object.

**4. What is the diﬀerence between an abstract class and an interface?**

An abstract class is a special type of class that cannot be instantiated, while the interface is basically a contract—it doesn’t have any implementation.

Also the abstract class at least one pure virtual function, and an interface can be simulated by making all methods as pure. (“Pure Virtual Functions and Abstract Classes in C++ - GeeksforGeeks”).

**5. Describe the following (make sure you compare and contrast as well):**

**Private** are items: variables or functions created inside a class that other classes cannot access directly unless I make these items public or allow some of my functions to change these variables inside that class.

**Public**: items like functions or variables are created inside a section of a class that other classes will be able to access and communicate with.

**Protected** can be accessed by inheritance classes. (Alsaud)

6. What is the diamond problem?

The diamond problem occurs when two superclasses of a class have a common base class. For example, in the following diagram, the TA class gets two copies of all attributes of Person class, this causes ambiguities. For example, consider the following program. (“Multiple Inheritance in C++ - GeeksforGeeks”)

7. Discuss Early and Late binding?

Early or static binding refers to events that occur at compile time, and late (or dynamic) binding is the opposite which refers to runtime binding. (badges)

8. Using a single variable, execute the show method in Base and in Derived. Of course you can use other statements as well, but only one variable.



|  |
| --- |
| #include <iostream>  using namespace std;  class Base{  public:  virtual void show() { cout<<" In Base n"; }  };  class Derived: public Base{  public:  void show() { cout<<"In Derived n"; }  };  int main(){  Base \*Pointer;  Derived DerObject;  Pointer = &DerObject;  Pointer -> show();  return 0;  } |
|  |

9. (15 points) Given the two class deﬁnitions below:

You need to write a deﬁnition for a Car class using the above two classes. You need to extend one, and use the other as a data member. This question boils down to composition vs inheritance. Explain your reasoning after your write you Car deﬁnition (bare bones deﬁnition).

#include <iostream>

#include <string>

using namespace std;

class Engine {

protected:

int horsepower;

public:

Engine(){

horsepower = 200;

cout << "Engine is running"

}

Engine(int \_horsepower){

\_horsepower = horsepower;

}

}; // The Engine class.

class Automobile: public Engine

{

protected:

string color;

string model;

string year;

public:

Automobile(){

year = "1900";

model = "msuCar";

color = "red";

}

~Automobile();

void turnRight(){}

void turnLeft(){}

void moveForward(){}

void moveBackward(){}

}; // Automobile class which is parent to Car class.

class Car: public Automobile // derived class from Automobile

{

protected:

Engine OneEngine;

public:

Car();

~Car();

};

int main() {

Car car1;

cout << "Hello World!\n";

}

10. (20 points) Write a class that contains two class data members numBorn and numLiving. The value of numBorn should be equal to the number of objects of the class that have been instanced. The value of numLiving should be equal to the total number of objects in existence currently (i.e., the objects that have been constructed but not yet destructed.)

#include <iostream>

using namespace std;

class MainClass // MainClass

{

protected:

int numBorn;

int numLiving;

public:

MainClass(){

numBorn += numBorn;

numLiving += numLiving;

}

~MainClass();

};

int main() {

cout << "Hello World!\n";

}

11. Write a program:

**(a) (10 points) Write a program that has an abstract base class named Quad. This class should have four member data variables representing side lengths and a pure virtual function called Area. It should also have methods for setting the data variables.**

**(b) (15 points) Derive a class Rectangle from Quad and override the Area method so that it returns the area of the Rectangle. Write a main function that creates a Rectangle and sets the side lengths.**

**(c) (10 points) Write a top-level function that will take a parameter of type Quad and return the value of the appropriate Area function.**

#include <iostream>

using namespace std;

class Quad // Quad class

{

protected:

int length;

int height;

public:

virtual int Area() = 0; // Pure virtual function called Area

Quad(){

length = 0;

height = 0;

}

void set(int \_x, int \_y){

length = \_x;

height = \_y;

}

}; // class Quad

class Rectangle: public Quad{ // Derive a class Rectangle from Quad

protected:

public:

int Area(){ //override the Area method so that it returns the area of the Rectangle

return length \* height;

}

};

//Write a top-level function that will take a parameter of type Quad and return the value of the appropriate Area function.

int getArea(Quad \*pt){

return pt->Area();

}

int main() {

Rectangle Rec;

Rec.set(3, 3);

cout << "The size of Rectable is" << getArea(&Rec);

return 0;

}

12. (10 points) What is the rule of three? You will have answered this question (in pieces) already, but in the OOP world, what does it mean?

Rule of three are Destructor, Copy Constructor, Copy Assignment Operator.

The rule of three is also known as the Law of Big Three or The Big Three and prescribes for class that, if a class defines any of the mentioned three then it should probably explicitly define all three −destructor, copy constructor, and copy assignment constructor. (“C++ Rule Of Three.”)

13. (10 points) What are the limitations of OOP

* Size: They are much larger than others.
* Effort: programs need a lot of work to create and build.
* Speed: they’re slower than other programs.

## Works Cited

## 

“13.15 — Overloading the Assignment Operator | Learn C++.” *Learn C++*, https://www.learncpp.com/cpp-tutorial/overloading-the-assignment-operator/. Accessed 28 Apr. 2021.

Alsaud, Fowzy. “2143-OOP-ALSAUD/Assignments/A05 at Main · Fowzy/2143-OOP-ALSAUD · GitHub.” *GitHub*, https://github.com/fowzy/2143-OOP-ALSAUD/tree/main/Assignments/A05. Accessed 28 Apr. 2021.

badges, Christian HagelidChristian Hagelid 7., 80944. “What Is the Difference between Early and Late Binding? - Stack Overflow.” *Stack Overflow*, https://stackoverflow.com/questions/10580/what-is-the-difference-between-early-and-late-binding. Accessed 28 Apr. 2021.

“C++ Class Constructor and Destructor - Tutorialspoint.” *RxJS, Ggplot2, Python Data Persistence, Caffe2, PyBrain, Python Data Access, H2O, Colab, Theano, Flutter, KNime, Mean.Js, Weka, Solidity*, https://www.tutorialspoint.com/cplusplus/cpp\_constructor\_destructor.htm. Accessed 28 Apr. 2021.

“C++ Rule Of Three.” *RxJS, Ggplot2, Python Data Persistence, Caffe2, PyBrain, Python Data Access, H2O, Colab, Theano, Flutter, KNime, Mean.Js, Weka, Solidity*, https://www.tutorialspoint.com/cplusplus-rule-of-three. Accessed 28 Apr. 2021.

“Copy Constructor in C++ - GeeksforGeeks.” *GeeksforGeeks*, 7 June 2014, https://www.geeksforgeeks.org/copy-constructor-in-cpp/.

“Default Constructors - Cppreference.Com.” *Cppreference.Com*, https://en.cppreference.com/w/cpp/language/default\_constructor. Accessed 28 Apr. 2021.

Griffin, Dr. Terry. “2143-Object-Oriented-Programming/CourseNotes.Pdf at Master · Rugbyprof/2143-Object-Oriented-Programming · GitHub.” *GitHub*, https://github.com/rugbyprof/2143-Object-Oriented-Programming/blob/master/CourseNotes.pdf. Accessed 28 Apr. 2021.

“Multiple Inheritance in C++ - GeeksforGeeks.” *GeeksforGeeks*, 21 Sept. 2012, https://www.geeksforgeeks.org/multiple-inheritance-in-c/.

“Pure Virtual Functions and Abstract Classes in C++ - GeeksforGeeks.” *GeeksforGeeks*, 15 July 2014, https://www.geeksforgeeks.org/pure-virtual-functions-and-abstract-classes/.

“Shallow Copy and Deep Copy in C++ - GeeksforGeeks.” *GeeksforGeeks*, 29 Dec. 2020, https://www.geeksforgeeks.org/shallow-copy-and-deep-copy-in-c/.

“Static vs Dynamic Typing | PERPETUAL ENIGMA.” *PERPETUAL ENIGMA*, https://www.facebook.com/WordPresscom, 4 Oct. 2014, https://prateekvjoshi.com/2014/10/03/static-vs-dynamic-typing/.

“Structure vs Class in C++ - GeeksforGeeks.” *GeeksforGeeks*, 10 Jan. 2011, https://www.geeksforgeeks.org/structure-vs-class-in-cpp/.