Student Name:_____

Learning other Programming Languages: *The Basics*

Moving from C++ to _____

List major differences, or syntax examples

Overall Structure of a Program • Where does execution begin? • File extensions • Global vs other • Importance of the order of functions	
Use of Whitespace/Statements • What indicates the end of one statement and the beginning of the next? • How does whitespace impact flow?	
Data Types • What types are available? • How strongly typed is the language? • Any differences between Primitive vs Other types? • How are constants supported?	
Using Instances of a Type • Example of the Syntax: • Do variables/objects have to be defined prior to being used?_____ • How are mixed types supported? • Identifier naming conventions: • Rules of initialization	
Operators Available • Assignment Syntax: • Arithmetic Operators: • Relational/Equality Operators: • Example of how they are used • Others	
Conditionals • Available Logical operators: • Syntax of “if”: • Support of “else” or “else if” concept?	

Loops • Available options: • Syntax of each (summarize any differences)	
Arrays • Are they available?_____ • How are they defined/created? • When is the size determined? • What is the Range of indices allowed? • Is there support for bounds checking? • Can they be dynamically allocated?_____ show syntax:	
Lists/Records/Structures • Which are available?_____ • What are their limitations? • How are they defined/created? • How are they used?	
Functions • Are functions first declared?_____ • How are they defined? • What type of return types allowed (any differences than expected)? • How are arguments specified? • Pass by reference vs Pass by value: • Is function overloading supported?____ • Are there constant arguments?____ ○ What does it mean?_____ • Show syntax example:	
I/O – Show a quick summary of each: • Displaying prompts • Displaying primitive types • Display other types, if different • Reading in primitive types • Reading in other types, if different • How to handle reading in multiple words for a phrase or sentence • Delimiter issues to know about?	

Student Name:_____

Learning other Programming Languages: *OOP and Advanced Concepts*

Moving from C++ to _____

List major differences, or syntax examples

Creating an Abstraction (e.g., Class) • Where is it placed: _____ • Overall syntax (differences) • Support for public, protected, private • Syntax of fields (e.g., data members): • Show Initialization of those fields: • Syntax for methods:	
Initializing and Abstraction • Syntax for "constructors" • How are arguments handled • What happens if a constructor? is not provided? • Can a constructor cause another constructor of the same class to be used? _____	
Dynamic Memory Support • How is dynamic memory allocated: • Are there destructors?_____ Syntax: • How is dynamic memory deallocated? o Is there a garbage collector?_____ • How are the addresses supported (pointers, references, etc.)_____	
Deep vs Shallow Issues • Is there a need for a copy constructor?____ • When would the copy constructor be invoked?_____ • How can objects be copied from one to another (such as with assignment)? • How can objects be compared to one another?	

Creating a Hierarchy • Syntax for specifying a sub-class vs super-class (e.g., derived vs base) • How is hiding supported within a hierarchy? (from one abstraction to another) • For what scope is function overloading supported (if any)	
Initializing in a Hierarchy • How can a sub-class cause a super-class constructor to be invoked? _____ (e.g., C++ initialization list concept)	
At the end of an Objects Lifetime • What does a sub-class need to do to cause a super-class function (like a destructor) to be invoked (e.g., such as functions like finalize in Java):	
Deep vs Shallow in a Hierarchy • What is the syntax needed when copying one sub-class object to another to make sure the entire object is copied? (e.g., C++ causing a base class = operator to be invoked) • What is the syntax needed when comparing one sub-class object to another to make sure the entire object (with all of its ancestor fields) is properly compared?	
Access Rules • Show the syntax for calling a method through an object:_____ • Is it possible for the client to call a super-class method that is hidden by a derived class method? _____ Show Syntax:	

Dynamic Binding • What are the Rules for dynamic binding? • What happens if the return types are not the same?_____ • What happens if the argument lists are not the same?_____ • What happens if upcasting is not used (base class reference/pointer referring to a derived class object)_____ • Can it be enabled/disabled?_____ • Show the syntax for the client calling a derived method using dynamic binding:	
Abstract Base/Super Class • Is it supported?_____ • Syntax (e.g., what is the replacement for C++'s pure virtual function) • Do all methods need to be abstract?_____ • Can the abstract class have fields?_____ • What happens if a sub-class (derived) doesn't implement one of the abstract methods?_____	
Other Constructs important to summarize (and may be technical interview topics!) • Templates/Generics • Operator Overloading • User Defined Type Conversion • Interfaces • Static keyword • Modules/Packages • Exception Handling • Important libraries/collections	