Career Services Assignment 3 – Java Flash Cards

Points possible: 50

Category	Criteria	% of Grade
Completeness	All requirements of the assignment are complete.	100

Instructions: Research common JavaScript interview questions online and create 20 flash cards from the information you find. Study your flash cards regularly to better prepare for interviews. Fill out the table below with the information you put on each of your flash cards.

Front of Card	Back of Card
JavaScript Interview Questions	Source: https://www.dotnettricks.com/learn/javascript/javascript-interview-questions
1. What is JavaScript?	Ans: JavaScript is an object-based programming language, mostly used as a client-side programming language with the HTML page to add some behavior to it. JavaScript was initially created as a browser-only language, but not it can be executed on the server or any client which has a JavaScript Engine. A product like Node.js, MongoDB, jaggery.js, ASP and many more use server-side JavaScript. In the browser, JavaScript can do many things as given below: # manipulating the HTML element
2. What is ECMAScript?	Ans. ECMAScript is a scripting language standardized by ECMA International-262. Languages like ActionScript, JavaScript, and many more scripting languages are used ECMAScript, among these JavaScript is a well known client-side language and an implementation of ECMAScript, since the standard was published. The latest version is ECMAScript6.
3. What are the data types supported by JavaScript?	Ans. JavaScript variables are dynamically typed, which means there is a data type but it will not be bound to a particular type. For example, while initializing the variable it can be string type, but later it can assign to a numeric value. There are two types of data types that are being supported which are primitive data types and non-primitive data types, below are some of the data types supported by JavaScript. The data types supported by JavaScript are: Undefined Null Boolean Object String Symbol Number
4. What is the difference between undefined and not defined?	Ans. Consider below example var x; console.log(x); Now in the console, we will get a message x is 'undefined' which means the variable is declared and memory is created but the value is not assigned to it. Console.log(y); In this case, you will get a message like 'not defined' because the variable y is not created, and memory is not allocated for it and we try to reference the variable.

5. What is the use of typeof operator?	Ans. The typeof is a unary operator which means it takes a single operand in a statement or expression, it is used to check the data type of its operand in the form of a string for example if we check the variable which is undefined then the typeof will return values as "undefined."
	
	var x=10;
	console.log(typeof (x));
	
	It will print the number in the console
	
	var x = 10;
	console.log(typeof(x) == 'number');
	
	From the above code if the typeof x is a number, so from the expression it will print true in the console.

6. What is the instanceof operator?	Ans. instanceof operator checks whether the object is an instance of a class or not.
	
	Function Country(name) {this.name = name};
	var country = new Country("India");
	console.log(country instanceof Country) // return true;
	
	It will also consider inheritance.
	
	Let arr = ['apple', 'orange', 'grapes'];
	console.log(arr instanceof Array); //prints true in console
	console.log(arr instanceof Object); // prints true in console
	
	arr is an array, but it also belongs to the object, because array prototypal inherits from the object.

7. What is the strict mode?	Ans. "use strict" is not a statement but a literal expression which is supported by ECMAScript version 5. This statement instructs the browser to use the strict mode, which is a safer future in JavaScript. It will remove some JavaScript silent errors.

	The strict mode applies to the entire script or to the individual functions and it doesn't apply to the block statements or close which is enclosed by the curly braces {}. Attempting to apply it to such contexts does not have any meaning. At multiple places such as eval code, functional code, event handler attributes, strings passed along with the setTimeout() and related functions are completely scripts, and invoking the strict mode in them works as expected to check the syntax vulnerabilities.
	
	Example
	
	"use strict";
	x = 10; //this will give error
	
	The above statement will give an error because in strict mode the variable should be declared before it is used.
	
	The "use strict" expression can be in global scope as well as local scope
	
	Global scope
	
	const employee = {name: "Ram", age: 25}
	employee.name = "Raju" // it is possible
	"use strict"
	x = 10; this will give error
	
	Local scope
	
	x = 10; // this will not give error.
	MyFunction();
	function myFunction() {
	  "use strict";
	y = 15; // this will give error
	}

8. Explain string in JavaScript?	Ans. The group of characters or textual data is called a string, in JavaScript, there is no separate type for the character, even a single character will be stored as a string. In JavaScript, the string can be enclosed with single quotes or double-quotes. But with JavaScript, the methods and properties are also available to primitive values, because JavaScript treats primitive values as an object when executing the methods and properties. Var str = "hello"; console.log(str); // print hello
9. What are the differences between search() and indexOf() ?	Ans. The differences between search() and indexOf() are given below: Search() : It is used to find a specified value and returns the position of the match, the value can be a string or a regular expression. var m = /e/;

	var str = "apple";
	str.search(m); // return 4
	
	--- 	
	
	indexOf() :
	It is used to find a specified value and returns the position of the match, the value should be a string, it won't accept a regular expression.
	
	var m = 'e';
	
	var str = "apple";
	str.indexOf(m); // return -1

10. What are the differences between indexOf() and lastIndexOf()?	Ans: The differences between indexOf() and lastIndexOf() methods are given below:
	
	indexOf() :
	It will return the index of the first occurrence of specific text in a string.
	
	var str = "Hello find me test me";
	str.indexOf("me"); // return 11
	
	--- 	
	
	lastIndexOf() :
	It will return the index of the last occurrence of specific text in a string.
	
	var str = "Hello find me test me";
	str.lastIndexOf("me"); // return 19

11. What are the differences between substr() and substring()?	Ans. The Differences between substr and substring methods are given below:
	
	substr() :
	It is used to return the characters in a string beginning at the specified index and returns the number of characters based on the length provided.
	
	var x = "hello";
	console.log((x.substr(1, 4) == "ello"));
	
	// It will print true in the log
	
	--- 	
	
	substring() :
	It is used to return the characters in a string beginning at the specified index and returns the number of characters based on length provided-1.
	
	var x = "hello";
	console.log((x.substring(1, 4) == "ello"));
	
	// It will print false in the log
	
	var x = "hello";
	console.log((x.substring(1, 5) == "ello")) // prints

	true in the console
12. What are the differences between an array and object?	Ans. The differences between array and object are given below: Array The array uses the numbered indexes to access the element in it; You should use an array when you want the element name to be a number; It is an ordered collection. Object The object uses the named indexes to access the members in it; You should use an object when you want the element name to be a string; It is a collection of unordered properties.

13. What is the self-executing function?	Ans. The self-executing function will execute right after it has been defined. The advantage of using it is, that it will execute the code without declaring any global [??]. Mostly it will be used to attach event listeners to DOM elements and other initialization work.
	
	This type of self-executing of function does not have its own name and hence it is called an anonymous function. The function has a trailing set of parenthesis without any arguments. The parameters for this function could be passed in the parenthesis.
	
	Below is a simple example showing the usage of the anonymous function:
	
	(function ()
	{
	// function body
	})();

14. What is the arrow function?	Ans. The arrow function will support in JavaScript only after ES6 or above, it is a short way to write function expressions. The conventional way of writing a function [?].
	
	The arrow function is basically a shorter syntax for using a function that does not have it's own "this", below is a simple example of the same:
	
	function add(a, b) {
	  return a + b;
	}
	console.log(add(1, 2)); // returns 3
	
	Using arrow function:
	
	add = (a, b) => {
	  return a + b;
	}
	console.log(add(1, 2)); // returns 3

15. How to find the browser which is running the web	Ans. The window object navigator is used to find the browser which is currently running the web

page?	application.
	
	var browserName = navigator.appName;
	console.log(browserName);

16. How to redirect the user to a new page?	Ans. We can use the window object location to redirect the user to the new page by providing the HREF URL link to be redirected to. Window.location.href="https://www.dotnettricks.com/"

17. What is the output of the below code?	var num = "10";
	(function () {
	console.log("Original Number " + num);
	var num = "50";
	console.log("New Number " + num);
	})();
	
	Ans. Original number undefined.
	
	New Number 50
	
	Reason: You will expect the original number will take the value from the outer scope, but the salary value was undefined, *because of hoisting.*

18. What is DOM?	Ans. DOM is a W3C (World wide web consortium) standard, when the HTML page loads in the browser, the browser creates the DOM (Document Object Model). It defines the HTML element as an object and allows scripts to dynamically manipulate the content, and the structure of the document. When any of the HTML documents are loaded in the browser, it will become a document object which is the root element that represents the HTML document. Each DOM element has various properties and methods, and with the help of document objects, we may add dynamic content to our web page according to the required behavior. HTML: <!DOCTYPE html> <html lang="en"> <body> <h1>Document Object Model</h1> </body> </html> In DOM, every HTML is an object, Nested tags are "children," the text inside a <h1> is an object as well. The DOM Tree of Objects The DOM represents HTML as a tree structure of tags. Here is how it looks in the browser "inspect the element." [no picture or words follow here.]
19. What is BOM?	Ans. BOM (Browser Object Model) provides interaction with the browser, the default object of the browser is a window. The various property provided by windows is a document, history, screen, location,

	and navigator. All the modern browsers have implemented the same methods and properties for JavaScript operational interactions which are often referred to as a BOM's methods and properties. A window object is automatically created by the browser itself.
20. What is the NaN property in JavaScript?	Ans. NaN property shows the "Not-a-Number" value. It shows a value that is not a legal number. One type of NaN would return a Number. If you want to check if a value is NaN, the isNaN() function is used. It is important to note that the isNaN() function transforms the given value to a Number type; later on, it equates to NaN.

21. What is the usefulness of the window object?	Ans. A browser's history object could be used to switch to history pages like back and forward from the existing page or another page. 3 methods of history object are as follows:
	
	1. history.back() // this method loads the previous page
	2. history.forward() // this method loads the next page
	3. history.go(number) // Its number may be positive (for forwarding) or negative (for backward). It will load the provided page number.

22. What is the working of timers in JavaScript?	Ans. Timers are useful to work a piece of code at a specific time or iterate the code in a specific interval. The same is performed by using functions like setInterval, setTimeout, and clearInterval. Timers are executed in a single thread. So, maybe queue up and there may be a waiting tme for execution. The setTimeout(function, delay) function is useful for starting a timer that calls a specific function after the stated delay. The setInterval(function, delay) function frequently operates the provided function in the stated delay and only stops when canceled. The timer gets to know when to stop with the clearInterval(id) function.
23. What are the various types of errors in JavaScript?	Ans. Here are the 3 types of errors in JavaScript: Runtime errors: These are the errors that occur due to misuse of the command within the HTML language. Load time errors: These errors occur while loading a web page. An example includes improper syntax that produce the errors dynamically. Logical errors: These errors come up because of the bad logic carried out on a function with a varied operation.
24. What is the "Strict Mode in JavaScript"? How can you enable it?	Ans. Strict Mode inserts some compulsions to JavaScript. In the strict Mode, JavaScript displays errors for a segment of code that did not display an error previously. However, it may be tricky and potentially insecure. Also, Strict Mode also resolves some errors that may obstruct the efficient working of the JavaScript engines. You can enable Strict Mode by inserting the string

	literal "use strict" above the file. Look at the following example to get a better idea:
	
	function myFunction() {
	"use strict";
	var v = "This shows implementation of strict mode function";
	}

25. Explain the difference between .call() and .apply()	Ans. The function .apply() and .call() are very identical in their usage but comes with a minor difference. The .call() is employed whenever a programmer knows the number of the function's arguments. This is because they have to be stated as arguments within the call statement. Conversely, .apply() is employed whenever the number is unknown. Also, this function .apply() needs that the argument should be an array. The key difference between these two functions is how the arguments are passed to the function.
26. How is DOM used in JavaScript?	Ans. DOM (Document Object Model) is accountable for how different objects in a model interrelate with each other. It is useful for developing web pages that contain objects like links, paragraphs, etc. Such objects can be executed to contain actions like add or delete. Furthermore, DOM is also useful to equip a web page with extra capabilities. The use of API provides a benefit compared to other prevailing models. If you deeply go through the JavaScript tutorial, you can know more about DOM.
27. What is the role of deferred scripts in JavaScript?	Ans. The parsing of HTML code during page loading is by default paused until the script has not halted executing. The webpage is delayed if the server is slow or the script is chiefly heavy. When using the Deferred, scripts would delay execution of the script until the HTML parser is operating. It decreases the web pages' loading time and they get showcased faster.
28. What are the different functional components in JavaScript?	Ans. Functional components are important topics covered in a JavaScript Course. Two type of functional components in JavaScript are – first class functions and nested functions. i. First class functions: these functions in JavaScript are used as first-class objects. Usually, this means that such functions can be passed in form of arguments to other functions. Also, they are returned as values from other functions or assigned to variables, or they can be saved in data structures. ii. Nested functions: Those functions that are defined within other functions are termed nested functions. Whenever the main function is invoked, nested functions are called.
29. What are the different ways to access the HTML elements in JavaScript?	Ans. The following DOM Methods are used to capture the HTML element and manipulate it: 1. getElementById('idname') - > this function is used

	to select the HTML element based on ID property of the HTML element.
	
	<!DOCTYPE html>
	<html>
	<head>
	<meta charset=”utf-8” />
	<title></title>
	</head>
	<body>
	<label id=”myelement”></label>
	<script>
	document.getElementById(‘myelement’).innerHTML = ‘ Welcome ’
	</script>
	</body>
	</html>
	
	2.
	getElementsByClassName(‘className’) - > This function is used to select the HTML elements based on the class name in DOM, it will return all matched HTML elements with respect to the class name.
	
	<!DOCTYPE html>
	<html>
	<head>
	<meta charset=”utf-8” />
	<title></title>
	<style>
	.lblMsg {
	color: #000;
	}
	</style>
	// does style go in head?
	</head>
	<body>
	<label id=”myelement” class=”lblMsg”></label>
	<script>
	document.getElementByClassName(“lblMsg”)[0].innerHTML = ‘ Welcome ’
	</script>
	</body>
	</html>
	
	3.
	getElementsByTagName(‘HTMLtagname’) - > This function is used to select the HTML elements based on the Tag name in the DOM, it will return all matched HTML elements with respect to the tag name.
	
	<!DOCTYPE html>
	<html>
	<head>
	<meta charset=”utf-8” />
	<title></title>
	</head>
	<style>
	.lblMsg {
	color: #000;

	} </style> <body> <label id=”myelement” class”lblMsg”></label> <script> document.getElementsByTagName(‘label’)[0].innerHTML = ‘ Welcome ’ </script> </body> </html>
30. [Personally written question; what are all the JavaScript functions you can remember off the top of your head?]	Ans. splice(), filter(), map(), isNaN(), indexOf(), lastIndexOf(),
C++ Interview Questions	Source: InterviewBit C++ Interview Questions downloaded pdf
1. What are the different data types present in C++?	Ans. The 4 data types in C++ are given below: --a-Primitive Datatype(basic datatype). Example- char, short, int, float, long, double, bool, etc. --b-Derived datatype. Example- array, pointer, etc. --c-Enumeration. Example- enum --d-User-defined data types. Example- structure, class, etc.
2. What is the difference between C and C++?	Ans. Main differences: C – procedure oriented language C++ - object oriented C – does not support data hiding C++ - Data is hidden by encapsulation to ensure that data structures and operators are used as intended. C – is a subset of C++ C++ - is a superset of C C – Function and operator overloading are not supported in C C++ - Function and operator overloading is supported in C++ C – Namespace features are not present in C C++ - Namespace is used by C++, which avoids name collisions C – Functions can not be defined inside structures C++ - Functions can be defined inside structures. C - calloc() and malloc() functions are used for memory allocation and free() function is used for memory deallocation C++ - new operator is used for memory allocation and deletes operator is used for memory deallocation.
3. What are class and object in C++?	Ans. A class is a user-defined data type that has data members and member functions. Data members are the data variables and member functions are the functions that are used to perform operations on these variables.

	An object is an instance of a class. Since a class is a user-defined data type so an object can also be called a variable of that data type.
	
	A class is defined as-
	
	class A{
	private:
	int data;
	public:
	void fun(){
	
	}
	};
	
	For example, the following is a class car that can have properties like name, color, etc. and they can have methods like speed().

4. What is the difference between struct and class?	Ans. In C++ a structure is the same as a class except for a few differences like security. The difference between struct and class are given below: Structure a: members of the structures are public by default Class a: members of the class are private by default Structure b: when deriving a struct from a class/struct, default access specifiers for base class/struct are publicly Class b: when deriving a class, default access specifiers are private.

5. What is operator overloading?	Ans. Operator Overloading is a very essential element to perform the operations on user-defined data types. By operator overloading we can modify the default meaning to the operators like +, -, *, /, <=, etc.
	
	For Example -
	The following code is for adding two complex number using operator overloading-
	
	class complex{
	private:
	float r, I;
	public:
	complex(float r, float i){
	this->r=r;
	this->i=i;
	}
	complex() {}
	void displaydata(){
	cout<<"real part = " <<r<<endl;
	cout<<"imaginary part = "<<i<<endl;
	}
	complex operator+(complex c){
	return complex(r+c.r, i+c.i);
	}

	};
	int main (){
	complex a(2, 3);
	complex b(3, 4);
	complex c=a+b;
	c.displaydata();
	return 0;
	}

6. What is polymorphism in C++?	Ans. Polymorphism in simple means having many forms. Its behavior is different in different situations. And this occurs when e have multiple classes that are related to each other by inheritance. For example, think of a base class called a car that has a method called car brand(). Derived classes of cars could be Mercedes, BMW, Audi – and they also have their own implementation of cars. The two types of polymorphism in C++ are: -Compile Time Polymorphism -Runtime Polymorphism

7. Explain constructor in C++	Ans. The constructor is a member function that is executed automatically whenever an object is created. Constructors have the same name as the class of which they are members so that the compiler knows that the member function is a constructor. And no return type is used for constructors.
	
	class A {
	private:
	int val;
	public:
	A(int x){
	val=x;
	}
	int main(){
	A a(3);
	
	return 0;
	}
	};

8. Tell me about virtual function.	Ans. Virtual function is a member function in the base class that you redefine in a derived class. A virtual function is declared using the virtual keyword. When the function is made virtual, C++ determines which function is to be invoked at the runtime based on the type of the object pointed by the base class pointer.
9. Compare compile time polymorphism and Runtime polymorphism	Ans. The main difference between compile-time and runtime is provided below: A. Compile-time polymorphism: In this method, we would come to know at compile time which method will be called. And the call is resolved by the compiler. A. Runtime polymorphism: In this method, we come to know at run time which method will be called. The call is not resolved by the compiler. B. Compile-time polymorphism: It provides fast execution because it is known at the compile time.

	B. Runtime polymorphism: It provides slow execution compared to compile-time polymorphism because it is known at the run time.
	
	C. Compile-time polymorphism: It is achieved by function overloading and operator overloading.
	C. Runtime polymorphism: It can be achieved by virtual functions and pointers.
	
	D. Compile-time polymorphism: Example -
	
	`int add(int a, int b){`
	`  return a+b;`
	`}`
	`int add (int a, int b, int c){`
	`  return a+b+c;`
	`}`
	`int main(){`
	`  cout<<add(2, 3)<<endl;`
	`  cout<<add(2, 3, 4)<<endl;`
	`  return 0;`
	`}`
	
	D. Runtime polymorphism: Example -
	
	`class A{`
	`  public:`
	`  virtual void fun(){`
	`  cout<<”base ”;`
	`  }`
	`};`
	`class B: public A{`
	`  public:`
	`  void fun(){`
	`  cout<<”derived ”;`
	`  }`
	`};`
	`int main(){`
	`  A *a=new B;`
	`  a->fun();`
	
	`  return 0;`
	`}`

10. What do you know about friend class and friend function?	Ans. A friend class can access private, protected, and public members of other classes in which it is declared as friends.
	
	Like friend class, friend function can also access private, protected, and public members. But, Friend functions are not member functions.
	
	For example -
	
	`class A{`
	`  private:`
	`  int data_a;`
	`  public:`
	`  A(int x){`

	`  data_a=x;`
	`  }`
	`  friend int fun(A, B);`
	`}`
	`class B{`
	`  private:`
	`  int data_b;`
	`  public:`
	`  A(int x){`
	`  data_b=x;`
	`  }`
	`  friend int fun(A, B);`
	`}`
	`int fun(A a, B b){`
	`  return a.data_a+b.data_b;`
	`}`
	`int main(){`
	`  A a(10);`
	`  B b(20);`
	`  cout<<fun(a, b)<<endl;`
	`  return 0;`
	`}`
	
	Here we can access the private data of class A and class B.

11. What are the C++ access specifiers?	Ans. In C++ there are the following access specifiers: Public: All data members and member functions are accessible outside the class. Protected: All data members and member functions are accessible inside the class and to the derived class. Private: All data members and member functions are not accessible outside the class.
12. Define inline function	Ans. If a function is inline, the compiler places a copy of the code of that function at each point where the function is called at compile time. One of the important advantages of using an inline function is that it eliminates the function calling overhead of a tradition function.

13. What is a reference in C++?	Ans. A reference is like a pointer. It is another name of an already existing variable. Once a reference name is initialized with a variable, that variable can be accessed by the variable name or reference name both.
	
	For example -
	
	`int x=10;`
	`int &ref=x; //reference variable`
	
	If we then change the value of ref it will be reflected in x. Once a reference variable is initialized it cannot refer to any other variable. We can declare an array of pointers, but an array of references is not possible.

14. What do you mean by	Ans. Abstraction is the process of showing the

abstraction in C++?	essential details to the user and hiding the details which we don't want to show to the user or hiding the details which are irrelevant to a particular user.
15. Is deconstructor overloading possible? If yes, then explain and if no then why?	Ans. No deconstructor overloading is not possible. Deconstructors take no arguments, so there's only one way to destroy an object. That's the reason destructor overloading is not possible.
16. What do you mean by call by value and call by reference?	Ans. In call by value method, we pass a copy of the parameter is passed to the functions. For these copied values a new memory is assigned and changes made to these values do not reflect the variable in the main function. In call by reference method, e pass the address of the variable and the address is used to access the actual argument used in the function call. So changes made in the parameter alter the passing argument.
17. What is an abstract class and when do you use it.	Ans. A class is called an abstract class whose objects can never be created. Such a class exists as a parent for the derived classes. We can make a class abstract by placing a pure virtual function in the class.

18. What are destructors in C++?	Ans. A constructor is automatically called when an object is first created. Similarly when an object is destroyed a function called destructor automatically gets called. A destructor has the same name as the constructor (which is the same as the class name) but is preceded by a tilde.
	
	Example:
	class AP{
	private:
	int val;
	public:
	A(int x){
	val=x;
	}
	A(){
	}
	-A(){ //destructor
	}
	}
	int main(){
	A a(3);
	return 0;
	}

19.What are the static members and static member functions?	Ans. When a variable in a class is declared static, space for it is allocated for the lifetime of the program. No matter how many objects of that class have been created, there is only one copy of the static member. So same static member can be accessed by all the objects of that class. A static member function can be called even if no objects of the class exist and the static function are accessed using only the class name and the scope resolution operator ::
20. Explain inheritance	Ans. Inheritance is the process of creating new classes, called derived classes, from existing classes. These existing classes are called base classes. The derived classes inherit all the capabilities of the base class but can add new features and refinements of their own. Example - Class Vehicle fuelAmount() capacity() applyBrakes() Class Bus Class Car Class Truck Class Bus, Class Car, and Class Truck inherit the properties of Class Vehicle. The most important thing about inheritance is that it permits code reusability.

21. What is a copy constructor?	Ans. A copy constructor is a member function that initializes an object using another object of the same class.
	
	Example -
	class A{
	int x,y;
	A(int x, int y){
	this->x=x;
	this->y=y;
	}
	};
	
	int main(){
	A a1(2, 3);
	A a2=a1; //default copy constructor is called
	return 0;
	}
	
	We can define our copy constructor. If we don't define a copy constructor then the default copy constructor is called.

22. What is the difference between shallow copy and deep copy?	Ans. The difference between shallow copy and a deep copy is given below: Shallow Copy A: Shallow copy stores the references of objects to the original memory address. Deep Copy A: Deep copy makes a new and separate copy of an entire object with its unique memory address. Shallow Copy B: Shallow copy is faster. Deep Copy B: Deep copy is comparatively slower. Shallow Copy C: Shallow copy reflects changes made to the new/copied object in the original object. Deep Copy C: Deep copy doesn't reflect changes made to the new/copied object in the original object.
23. What is the difference between virtual functions and pure virtual functions?	Ans. A virtual function is a member function in the base class that you redefine in a derived class. It is declared using the virtual keyword.

	Example -
	
	class base{
	public:
	  virtual void fun(){
	
	  }
	};
	
	A pure virtual function is a function that has no implementation and is declared by assigning 0. It has no body.
	
	Example -
	
	class base{
	public:
	  virtual void fun()=0;
	};
	
	Here, = sign has got nothing to do with the assignment, and value 0 is not assigned to anything. It is used to simply tell the compiler that a function will be pure and it will not have any body.

24. If class D is derived from a base class B, when creating an object of type D in what order would the constructors of these classes get called?	Ans. The derived class has two parts, a base part, and a derived part. When C++ constructs derived objects, it does so in phases. First, the most-base class (at the top of the inheritance tree) is constructed. Then each child class is constructed in order until the most-child class is constructed last. So the first constructor of class B will be called and then the constructor of class D will be called. During the destruction exactly reverse order is followed. That is destructor starts at the most-derived class and works its way down to base class. So the first destructor class of class D will be called and then the destructor of class B will be called.

25. Can we call a virtual function from a constructor?	Ans. Yes, we can call a virtual function from a constructor. But the behavior is a little different in this case. When a virtual function is called, the virtual call is resolved at runtime. It is always the member function of the current class that gets called. That is the virtual machine doesn’t work within the constructor.
	
	For example -
	
	class base{
	  private:
	  int value;
	  public:
	  base(int x){
	  value=x;
	  }

	virtual void fun(){
	
	  }
	
	class derived{
	  private:
	  int a;
	  public:
	  derived(int x, int y):base(x){
	  base *b;
	  b=this;
	  b->fun(); //calls derived::fun()
	}
	  void fun(){
	  cout<<”fun inside derived class”<<endl;
	  }
	}

26. What are void pointers?	Ans. A void pointer is a pointer which is having no datatype associated with it. It can hold addresses of any type.
	
	For example-
	
	void *ptr;
	char *str;
	p=str; // no error
	str=p; // error because of type mismatch
	
	We can assign a pointer of any type to a void pointer but the reverse is not true unless you typecast it as
	
	str=(char*) ptr;

27. What is this pointer in C++?	Ans. The member functions of every object have a pointer named this, which points to the object itself. The value of this is set to the address of the object for which it is called. It can be used to access the data in the object it points to.
	
	Example -
	
	class A{
	  private:
	  int value;
	  public:
	  void setvalue(int x){
	  this->value=x;
	  }
	};
	
	int main(){
	  A a;
	  a.setvalue(5);
	  return 0;
	}

28. How do you allocate and deallocate memory in C++?	Ans. The new operator is used for memory allocation and deletes operator is used for memory deallocation in C++. For example -

	int value=new int;
	
	// allocates memory for storing 1 integer
	
	delete value;
	
	// deallocates memory taken by value
	
	int *arr=new int[10];
	
	// allocates memory for storing 10 intended
	
	delete []arr;
	
	// deallocates memory occupied by arr

Linux Admin Interview Questions	Source: https://www.linuxtechi.com/experience-linux-admin-interview-questions/ next time should use this one: Source: https://www.whizlabs.com/blog/top-linux-interview-questions-answers/
1. What is Linux and also explain the basic components of Linux?	Answer: Linux is the most commonly used operating system that is open source and free. For any computer, the operating system acts as the backbone, and it is most important software that is required for any computer. From network routers, television, video games console, smartwatches, smartphones, desktops, laptops to any other electronic device, Linux is everywhere. Linux operating system is consist of 3 components which are as below: Kernel: Linux is a monolithic kernel that is free and open source software that is responsible for managing hardware resources for the users. System Library: System Library plays a vital role because application programs access Kernels feature using system library. System Utility: System Utility performs specific and individual level tasks.
2. What are the differences between UNIX and Linux Operating System?	Answer: To understand the differences between UNIX and Linux Operating system, first of all, we should know that Linux is a UNIX clone, the Kernel of which is created by Linus Torvalds. There are so many differences between Linux and UNIX operating system which are as follows: Open Source Operating System: The most significant difference between UNIX and Linux operating system is Linux is an open source operating system. The open-source operating system that means Linux source code is available for use so that developers can modify it as per their requirement. But UNIX operating system doesn't come under the broad category of an open-source operating system for which developers can edit it.

	Free of Cost: One of the biggest reason that it is broadly used is Linux operating system is free of cost. Linux operating system is free, but UNIX Operating system is not free. We can download it from the internet. Compatibility and Flexibility: If we compare the flexibility and compatibility of both operating system, you will find that Linux is more flexible than UNIX operating system and more compatible with different types of hardware as compared to UNIX operating System.
3. Describe BASH.	Answer: BASH stands for Bourne Again Shell. BASH is the UNIX shell for the GNU operating system. So, BASH is the command language interpreter that helps you to enter your input, and so you can retrieve information. In a straightforward language, we can say that it is a program that will understand the data entered by the user and execute the command and gives output.
4. What is crontab and explain its functionality and explain the format of crontab?	Answer: Cron is a scheduler that executes the commands at a regular interval as per the specific date and time defined. We have multiple users in Linux, and all the users can have their crontab separately. The crontabs files are saved at a particular location that is /var/spool/cron/crontabs. There are six fields in the format for the crontab that is as below: <Minute><Hour><Day_of_the_Month><Month_of_the_Year><Day_of_the_Week><command/program to execute>
1. Why LVM is required ?	Ans: LVM stands for Logical Volume Manager, to resize file system's size online we require LVM partition in Linux. Size of LVM partition can be extended and reduced the lvextend and lvreduce commands respectively.
2. How to check memory stats and CPU stats?	Ans: Using 'free' & 'vmstat' command we can display the physical and virtual memory statistics respectively. With the help of 'sar' command we see the CPU utilization & other stats.
3. What does Sar provides and at which location Sar logs are stored?	Ans: Sar collect, report, or save system activity information. The default version of the sar command (CPU utilization report) might be one of the first facilities the user runs to begin system activity investigation, because it monitors major system resources. If CPU utilization is near 100 percent (user + nice + system), the workload sampled is CPU-bound. By default log files of Sar command is located at /var/log/sa/sadd/ file, where the dd parameter indicates the current day.
4. How to increase the size of LVM partition?	Ans: Below are the Logical Steps: # use the lvextend command (lvextend -L +100M

	/dev/<Name of the LVM Partition> , in this example we are extending the size by 100MB.) # resize2fs /dev/<Name of the LVM Partition> -check the size of partition using df command
5. How to reduce or shrink the size of LVM partition?	Ans: Below are the logical steps to reduce size of LVM partition: # Unmount the filesystem using umount command [umount or unmount?] # Use resize2fs command, e.g. resize2fs /dev/mapper/myvg-mylv 10G # Now use the lvreduce command, e.g. lvreduce L 10G dev/mapper/myvg-lv Above Command ^ will shrink the file system and make the filesystem size 10GB.
6. How to create partition on the raw disk?	Ans: Using fdisk utility we can create partitions on the raw disk. Below are the steps to create partition: # fdisk dev/hd* (IDE) or dev/sd* (SCSI) # Type n to create a new partition. # After creating partition, type w command to write the changes to the partition table. # Type 'partprobe' to instruct the kernel to re-read the partition table.
7. Where are the kernel modules located?	Ans: The '/lib/modules/kernel-version/' directory stores all kernel modules or compiled drivers in Linux operating system. Also with 'lsmod' command we can see all the installed kernel modules.
8. What is umask?	Ans: umask stands for 'User file creation mask,' which determines the settings of a mask that controls which file permissions are set for files and directories when they are created.
9. How to set the umask permanently for a user?	Ans: To set this value permanently for a user, it has to be put in the appropriate profile which depends on the default shell of the user.
10. How to Boot RHEL / Rocky Linux / CentOS in Single User Mode?	Ans: Follow the beneath steps to boot RHEL / Rocky Linux / CentOS in single user mode: # Reboot and go to the grub prompt # Go to the end of line which starts with 'linux' and type 'rd.break' and hit enter. # Mount the root file system in rw mode and then do chroot/sysroot. # Perform the troubleshooting
11. How to share a directory using nfs	Ans: To share a directory using nfs, # First edit the configuration file '/etc/exportfs', add a entry like # /<directory-name> <ip or Network>(Options) # Restart the nfs service or 'exportfs -arv'
12. How to check and mount nfs share?	Ans: Using 'showmount' command we can see which directories are shared via nfs e.g. 'showmount -e <ip address of nds server>' Using mount command we can mount the nfs share on Linux machine.
13. What are the default ports used for SMTP, DNS, FTP, DHCP, SSH and squid?	Ans: Default ports are listed below: # Service | Port # SMTP 25 # DNS 53 # FTP 20 (data transfer), 21 (Connection established) # DHCP 67/UDP(dhcp server), 68/UDP(dhcp client) # SSH 22 # Squid 3128

14. What is Network Bonding?	Ans: Network bonding is the aggregation of multiple lan cards into a single bonded interface to provide fault tolerance and high performance. Network bonding is also known as NIC Teaming.
15. What are the different modes of Network bonding in Linux?	Ans: Below are the list of modes used in network bonding: # balance-rr or 0 – round-robin mode for fault tolerance and load balancing # active-backup or 1 – Sets active-backup moder for fault tolerance # balance-xor or 2 – Sets an XOR (exclusive-or) mode for fault tolerance and load balancing. # broadcast or 3 – Sets a broadcast mode for fault tolerance. All transmissions are sent on all slave interfaces. # 802.3ad or 4 – Sets an IEEE 802.3ad dynamic link aggregation mode. Creates aggregation groups that share the same speed & duplex settings. # balance-tlb or 5 – Sets a Transmit Load Balancing (TLB) mode for fault tolerance & load balancing. # balance-alb or 6 – Sets and Active Load Balancing (ALB) mode for fault tolerance & load balancing.
16. How to check and verify the status of the bond interface?	Ans: Using the command 'cat /proc/net/bonding/bond0', we can check which mode is enabled and what lan cards are used in this bond. In this example we have only one bond interface but we can have multiple bond interfaces like bond1, bond2 and so on.
17. How to check default route and routing table?	Ans: Using the commands 'netstat -nr', 'ip route show' and 'route -n' we can see the default route and routing tables.
18. How to check which ports are listening in my Linux Server?	Ans: With the help of 'ss', 'netstat -listen' and 'lsof -i' commands we can check ports listening status.
19. What is default data directory for docker containers?	Ans: Default data directory for docker containers are '/var/lib/docker'.
20. What is the difference between Docker and Podman?	Ans: Docker is a daemon based process, it means containers will only work when docker daemon is running, whereas Podman is daemon-less, it means containers don't need any daemon to run.
21. How to upgrade Kernel in Linux?	Ans: We should never upgrade Linux Kernel, always install the new kernel using dnf (or yum) or rpm command because upgrading a kernel can make your linux box in an unbootable state.
22. How to scan newly assigned luns on Linux box without rebooting?	Ans: There are two ways to scan newly assigned luns: # Method 1: if sg3 rpm is installed, then run the command 'rescan-scsi-bus.sh' # Method 2: Run the Command, 'echo "---" > /sys/class/scsi_host/hostX/scan'
23. How to find WWN numbers of HBA cards in Linux Server?	Ans: We can find the WWN numbers of HBA cards using the command 'systool -c fc_host -v | grep port_name'
24. How to add & change the Kernel parameters?	Ans: To set the kernel parameters in linux, first edit the file '/etc/sysctl.conf' after making the changes save the file and run the command 'sysctl -p', this command will make the changes permanently without rebooting the machine.
25. What is Puppet Server?	Ans: Puppet is an open-source & enterprise software

	server for configuration management toll in UNIX like operating system. Puppet is an IT automation software used to push configuration to its clients (puppet agents) using code. Puppet code can do a variety of tasks from installing new software, to check file permissions, or updating user accounts & lots of other tasks.
26. What are manifests in Puppet?	Ans: Manifests in Puppet are the files in which the client configuration is specified.
27. Which Command is used to sign requested certificates in Puppet Server?	Ans: ‘puppetca -sign hostname-of-agent’ in (2.X) & ‘puppet ca sign hostname-of-agent’ in (3.X)
28. How and Where to use Ansible ad-hoc commands?	Ans: Use the following syntax to use ansible ad-hoc command: # $ ansible [pattern] -m [module] -a <module option> # $ ansible webservers -m shell -a ‘df -Th’ Ad hoc commands are used for performing quick tasks and tests. We don’t need to write any playbook to run ad-hoc commands on ansible hosts.
29. How to find all the files under /var whose size is more than 200MB?	Ans: With the help of find command, we can list all the filds whose size is more than 200MB, # $ sudo find /var -type f -size +100M -exec ls -lah {} \;
30.What is load average in Linux?	Ans: Load Average is defined as the average sum of the number of processes waiting in the run queue and number of processes currently executing over the period of 1, 5, and 15 minutes. Using the ‘top’ and ‘uptime’ command we find the load average of a Linux server.