TSA Coding Study Cards 2

Study online at https://quizlet.com/_d1ci5p

1.	**Turning Machine**	a mathematical model of a hypothetical computing machine that uses a tape with symbols that can be read, written, or erased; the tape is infinite in length and divided into discrete cells that are read by a read/write head. The machine can read and write symbols on the tape and move the read/write head back and forth along the tape.
2.	**CPU (Central Processing Unit)**	the primary component of a computer that carries out the instructions of a computer program. It performs arithmetic and logical operations, manages the flow of data between different components of a computer, and communicates with input/output devices.
3.	**Transistor**	a semiconductor device that is used to amplify or switch electronic signals. Transistors can be used in various electronic devices, including computers, radios, and televisions.
4.	**Bit**	the smallest unit of information in computing, representing a binary digit that can have a value of either 0 or 1.
5.	**Byte**	a unit of digital information that typically consists of eight bits. Bytes are commonly used to represent characters, such as letters and numbers.
6.	**Character Encoding ASCII**	a character encoding system that represents each character in the English alphabet as a unique binary code, allowing computers to store and transmit text in a standardized format.
7.	**Binary**	a system of numerical notation that uses only two digits (0 and 1). Binary is commonly used in computing to represent data and instructions.
8.	**Hexadecimal**	a system of numerical notation that uses 16 digits, including 0-9 and A-F. Hexadecimal is often used in computing to represent binary data in a more human-readable format.
9.	**Nibble**	

		a unit of digital information that typically consists of four bits.
10.	**Machine Code**	low-level instructions that are directly executed by a computer's CPU. Machine code is typically written in binary or hexadecimal notation and is specific to a particular type of CPU.
11.	**RAM (Random Access Memory)**	a type of computer memory that allows data to be accessed in any order, regardless of its physical location in memory. RAM is used to store data and instructions that are currently being used by the CPU.
12.	**Memory Address**	a unique identifier for a specific location in computer memory. Memory addresses are used to access and manipulate data stored in memory.
13.	**I/O (Input/Output)**	the transfer of data between a computer and its input/output devices, such as a keyboard, mouse, monitor, or printer.
14.	**Kernel (Drivers)**	the core component of an operating system that manages the computer's resources and provides a layer of abstraction between hardware and software. Drivers are software components that allow the operating system to communicate with hardware devices, such as printers and scanners.
15.	**Shell**	a command-line interface that allows users to interact with a computer's operating system. Shells provide a way to execute commands, manipulate files and directories, and automate tasks.
16.	**Command Line Interface**	a text-based interface that allows users to interact with a computer's operating system by typing commands. Command line interfaces are often used by programmers and system administrators to automate tasks and perform complex operations.
17.		

	SSH (Secure Shell)	a network protocol that provides secure communication between computers. SSH is commonly used to remotely access and control computers over a network.
18.	**Mainframe**	a large, powerful computer that is designed to handle complex computing tasks and high volumes of data. Mainframes are commonly used in enterprise environments for mission-critical applications, such as financial transaction processing.
19.	**Programming Language**	a formal language that is used to write computer programs. Programming languages can be classified as low-level or high-level, depending on the level of abstraction they provide.
20.	**Abstraction**	In computer science, abstraction is the concept of hiding implementation details while showing only the necessary information to the user. It helps users to manage complexity and avoid unnecessary details, making it easier to use complex systems. For example, programming languages provide an abstraction layer for developers, hiding low-level details of computer hardware.
21.	**Interpreted**	Interpreted languages are programming languages where the source code is not directly compiled to machine code but is interpreted line by line at runtime by a software program known as an interpreter. The interpreter reads the source code and executes it in real-time, translating each line of code into machine instructions as it runs. Examples of interpreted languages include Python, Ruby, and JavaScript.
22.	**Compiled**	Compiled languages are programming languages where the source code is compiled into machine code before execution. A compiler takes the source code and converts it into machine code that can be directly executed by a computer's CPU. Examples of compiled languages include C, C++, Java, and Go.
23.	**Executable**	

		An executable is a file that contains machine code and can be executed directly by a computer's operating system. It typically has an .exe extension on Windows operating systems. Executables can be either compiled code or interpreted code that has been packaged in a standalone format.
24.	**Data Types**	Data types are the types of values that can be stored in a computer's memory. They include integer, floating-point, character, and string data types, among others. Different programming languages have different data types, and the type of data being used determines what kind of operations can be performed on it.
25.	**Variable**	A variable is a named storage location in a computer's memory that can hold a value. It is a fundamental concept in programming and is used to store and manipulate data. Variables can be of different data types and can hold different values at different points in time during program execution.
26.	**Dynamic Typing**	Dynamic typing is a feature of some programming languages where the data type of a variable is determined at runtime. In dynamically typed languages, a variable can hold any type of data, and the type is determined when the program is run.
27.	**Static Typing**	Static typing is a feature of some programming languages where the data type of a variable is determined at compile time. In statically typed languages, the type of a variable is specified explicitly when it is declared, and the compiler checks that the variable is being used correctly.
28.	**Pointer**	A pointer is a variable that stores the memory address of another variable. Pointers are used to manipulate and access data in memory, and they are a fundamental concept in programming.
29.	**Garbage Collector**	A garbage collector is a program that automatically frees up memory that is no longer in use by a program. It

		detects and removes memory that is no longer referenced by any part of the program, making it easier for developers to write memory-efficient code.
30.	**int**	In programming, int is a data type that stands for "integer." It is used to store whole numbers, both positive and negative.
31.	**signed/unsigned**	In programming, signed and unsigned refer to the way that integers are stored in memory. Signed integers can store both positive and negative values, while unsigned integers can only store positive values.
32.	**float**	In programming, float is a data type that stands for "floating-point." It is used to store decimal numbers.
33.	**Double**	Double is a data type that stands for "double-precision floating-point." It is similar to the float data type but can store more precise decimal numbers.
34.	**Char**	In programming, char is a data type that stands for "character." It is used to store single characters, such as letters, numbers, and symbols.
35.	**String**	In programming, a string is a sequence of characters. It is used to represent text and is a fundamental data type in many programming languages.
36.	**Big endian**	A type of byte ordering in which the most significant byte (MSB) is stored in the smallest memory address and the least significant byte (LSB) is stored in the largest memory address.
37.	**Little endian**	A type of byte ordering in which the least significant byte (LSB) is stored in the smallest memory address and the most significant byte (MSB) is stored in the largest memory address.
38.	**Array**	A data structure that stores a collection of elements of the same type in a contiguous block of memory, allowing

		for efficient access to individual elements based on their index.
39.	**Linked List**	A data structure that consists of a sequence of elements, where each element points to the next one, allowing for efficient insertion and deletion of elements in the middle of the list.
40.	**Set**	A collection of unique elements that does not allow duplicates, typically implemented using a hash table or a balanced tree data structure.
41.	**Stack**	A data structure that allows for adding and removing elements in a last-in-first-out (LIFO) order, where the most recently added element is always the first one to be removed.
42.	**Queue**	A data structure that allows for adding and removing elements in a first-in-first-out (FIFO) order, where the oldest element is always the first one to be removed.
43.	**Hash**	A function that takes in an input (usually a string or a number) and produces a fixed-size output (a hash value) that represents the input in a way that is difficult to reverse, typically used for indexing data in a hash table.
44.	**Tree**	A data structure that consists of a collection of nodes, where each node has a parent and zero or more children, typically used to represent hierarchical relationships between elements.
45.	**Graph**	A data structure that consists of a collection of nodes (vertices) and edges that connect them, typically used to represent relationships between arbitrary objects.
46.	**Nodes and Edges**	The basic building blocks of a graph, where nodes represent the vertices of the graph and edges represent the connections between them.
47.	**Algorithms**	

		A sequence of steps used to solve a particular problem or perform a particular task, typically involving input data, intermediate computations, and output results.
48.	**Functions**	A named sequence of statements that performs a specific task and can be called multiple times from different parts of a program, typically taking one or more input arguments and producing an output value.
49.	**Return**	A statement that terminates a function and returns a value to the caller.
50.	**Arguments**	Values passed into a function when it is called, which can be used to customize the behavior of the function and produce different output values.
51.	**Operators**	Symbols that represent specific operations or actions to be performed on one or more values, such as addition, multiplication, comparison, or logical operations.
52.	**Boolean**	A data type that represents one of two values, typically denoting true or false, or 1 or 0.
53.	**Expression**	A combination of values, variables, and operators that can be evaluated to produce a result.
54.	**Statement**	A unit of code that performs a specific action or operation, such as declaring a variable, assigning a value, or calling a function.
55.	**Conditional Logic**	A programming construct that allows for executing different code blocks based on the value of a particular condition, typically using if-else statements or switch statements.
56.	**While Loop**	A programming construct that allows for executing a block of code repeatedly while a particular condition is true, typically using a while statement.
57.	**For Loop**	

		A programming construct that allows for executing a block of code repeatedly for a fixed number of times, typically using a for statement.
58.	**Iterable**	An object that can be iterated over using a loop or other iteration construct, typically implemented using a collection data structure such as an array or a list.
59.	**Void**	In programming, void refers to the absence of a specific type. It is often used to indicate that a function does not return a value.
60.	**Recursion**	A process in which a function calls itself directly or indirectly. Recursion is often used in programming to solve problems that can be broken down into smaller subproblems.
61.	**Call Stack**	A data structure used by a computer program to store information about the active subroutines or functions. When a function is called, its information is pushed onto the stack, and when it returns, its information is popped off the stack.
62.	**Stack Overflow**	A condition in which the call stack exceeds its maximum size, causing the program to crash. This can occur when a function calls itself too many times or when a program is written in a way that causes excessive recursion.
63.	**Base Condition**	The condition in a recursive function that stops the recursion. Without a base condition, a recursive function can call itself indefinitely.
64.	**Big-O**	A mathematical notation used to describe the time complexity of an algorithm. It represents the upper bound of the worst-case scenario for the time complexity.
65.	**Time Complexity**	A measure of the amount of time it takes to run an algorithm, often expressed in terms of the size of the input data.

66.	**Space Complexity**	A measure of the amount of memory used by an algorithm, often expressed in terms of the size of the input data.
67.	**Brute Force**	A straightforward approach to solving a problem by considering all possible solutions and selecting the one that meets the requirements. Brute force algorithms can be very slow for large problems.
68.	**Divide and Conquer**	A technique in which a problem is divided into smaller subproblems that are solved separately. The solutions to the subproblems are combined to create a solution to the original problem.
69.	**Dynamic Programming**	A method for solving complex problems by breaking them down into smaller subproblems and solving each subproblem only once. The solutions to the subproblems are stored in a table and used to solve the larger problem.
70.	**Memoization**	A technique in which the results of expensive function calls are stored and returned when the same inputs occur again.
71.	**Greedy**	A strategy in which the best choice is made at each step of a problem without looking ahead. Greedy algorithms can be fast, but they may not always produce the optimal solution.
72.	**Dijkstra's Shortest Path**	An algorithm for finding the shortest path between two nodes in a graph with non-negative edge weights.
73.	**Backtracking**	A technique for solving problems by trying different solutions and backing up when a solution fails.
74.	**Declarative**	A style of programming in which the code describes the problem to be solved, rather than describing the steps needed to solve it.
75.	**Functional Language**	

		A programming language that emphasizes the use of functions to solve problems, often with a focus on immutability and the avoidance of side effects.
76.	**Imperative**	A style of programming that emphasizes the use of statements that change a program's state.
77.	**Procedural Language**	A programming language that emphasizes the use of procedures, or subroutines, to organize a program's logic.
78.	**Multiparadigm**	A programming language that supports multiple paradigms, such as functional, imperative, and object-oriented.
79.	**OOP**	Object-oriented programming, a programming paradigm that emphasizes the use of objects to represent data and behavior.
80.	**Class**	A blueprint for creating objects in object-oriented programming.
81.	**Properties**	Attributes of an object in object-oriented programming.
82.	**Methods**	Functions associated with an object in object-oriented programming.
83.	**Inheritance**	A mechanism in object-oriented programming that allows a class to inherit properties and methods from another class.
84.	**Design Patterns**	Reusable solutions to common programming problems.
85.	**Instantiate**	To create an object from a class in object-oriented programming.
86.	**Heap Memory**	The area of a computer's memory where dynamic memory allocation takes place. The heap is a data structure that allows the allocation and deallocation of memory at runtime. Unlike the stack, which is used for static memory

		allocation, the heap allows for dynamic memory allocation that can be used throughout a program's execution.
87.	**Reference**	A reference is an address or a pointer to a location in memory where a value or an object is stored. In programming, references are often used to pass objects between functions or to allow different parts of a program to access and modify the same data.
88.	**Threads**	A thread is a sequence of instructions within a program that can be executed independently of other threads. A program can have multiple threads running simultaneously, each performing a different task.
89.	**Parallelism**	Parallelism is the concept of dividing a task into smaller parts that can be executed simultaneously on multiple processors or cores. By using parallelism, a program can take advantage of multiple processors or cores to perform a task faster.
90.	**Concurrency**	Concurrency refers to the ability of a program to handle multiple tasks at the same time, even if they are not executed simultaneously. Concurrency can be achieved through the use of threads, processes, or other mechanisms that allow different parts of a program to execute independently.
91.	**Bare Metal**	Bare metal refers to a computer system that is running directly on hardware, without an operating system or other software layer. Bare metal systems are often used for high-performance computing, embedded systems, and other specialized applications.
92.	**Virtual Machine**	A virtual machine is an emulated computer system that runs on top of a host operating system. Virtual machines are often used to run multiple operating systems or to test software in a controlled environment.
93.	**IP Address**	An IP address is a unique identifier that is assigned to every device on a network that uses the Internet Protocol

		for communication. IP addresses are used to route data packets across a network and to identify the source and destination of network traffic.
94.	**URL**	A URL, or Uniform Resource Locator, is a string of characters that identifies the location of a resource on the internet. A URL typically consists of a protocol identifier, a domain name or IP address, and a path to the resource.
95.	**DNS**	DNS, or Domain Name System, is a system that translates human-readable domain names into IP addresses. DNS allows users to access websites and other resources on the internet using easy-to-remember domain names, rather than numeric IP addresses.
96.	**TCP**	TCP, or Transmission Control Protocol, is a protocol used for communication between devices on a network. TCP provides a reliable, ordered, and error-checked delivery of data between applications running on different devices.
97.	**Packets**	A packet is a unit of data that is transmitted across a network. Packets contain information about the source and destination of the data, as well as the data itself.
98.	**SSL**	SSL, or Secure Sockets Layer, is a security protocol that provides encryption and authentication for data transmitted over the internet. SSL is often used to secure online transactions and other sensitive communications.
99.	**HTTP**	HTTP, or Hypertext Transfer Protocol, is a protocol used for communication between web servers and web clients. HTTP defines the format of messages sent between the client and the server, as well as the actions that can be taken by each party.
100.	**API**	API, or Application Programming Interface, is a set of guidelines and standards for accessing a web-based software application or web tool. APIs provide a way for different software systems to communicate and share data with each other.

101. **Printers**	A printer is a device that produces hard copies of electronic documents or images. Printers can be connected to a computer directly or accessed over a network, and can produce output in a variety of formats, including text, graphics, and photographs.