**Basic Syntax and Data Types**

1. Write a Go program to print "Hello, World!".
2. How do you declare a variable in Go?
3. Create a program to swap two numbers without using a third variable.
4. Explain and implement constants in Go.
5. Write a program that reads input from the user.
6. Implement a basic calculator using switch-case.
7. Write a function to return the sum of two integers.
8. Write a Go function that checks if a number is even or odd.
9. What is the difference between := and var? Show with code.
10. Write a program to print numbers from 1 to 10 using a loop.

**🔹 Control Structures**

1. Write a program to find the largest of three numbers.
2. Implement a program to print the Fibonacci series up to n terms.
3. Write a program to check if a number is prime.
4. Write a program to check if a string is a palindrome.
5. Reverse a string without using the in-built reverse function.
6. Count vowels and consonants in a given string.
7. Find the factorial of a number using recursion.
8. Use a for loop to print a pattern (e.g., pyramid or triangle).
9. Use nested loops to print a multiplication table.
10. Implement break and continue with examples.

**Arrays and Slices**

1. Declare and initialize an array with 5 elements.
2. Write a program to find the maximum element in an array.
3. Calculate the sum of all elements in a slice.
4. Append elements to a slice dynamically.
5. Remove a specific element from a slice.
6. Sort a slice of integers in ascending order.
7. Merge two slices into one.
8. Reverse an array or slice in-place.
9. Copy one slice into another.
10. Find the frequency of elements in a slice.

**🔹 Maps and Strings**

1. Create a map with employee names and ages. Print them.
2. Check if a key exists in a map.
3. Count the frequency of characters in a string using a map.
4. Convert a string to a slice of runes and reverse it.
5. Compare two strings ignoring case sensitivity.
6. Remove duplicate characters from a string.
7. Find the first non-repeating character in a string.
8. Concatenate multiple strings efficiently.
9. Check if two strings are anagrams.
10. Replace all spaces in a string with %20.

**🔹 Functions and Recursion**

1. Write a recursive function to calculate power of a number.
2. Create a variadic function that takes multiple integers and returns their sum.
3. Implement a function that returns multiple values.
4. Write a function to reverse digits of an integer.
5. Use defer to track the order of function calls.
6. Create and call an anonymous function.
7. Write a higher-order function that accepts a function as argument.
8. Explain and show closures in Go.
9. Create a recursive function to compute GCD.
10. Create a tail-recursive function for factorial.

Structs & Methods (51–60)

1. Define a struct named Person with name and age fields.
2. Write a method for a struct to display its contents.
3. Create a constructor function for a struct.
4. Create and compare two struct instances.
5. Create an embedded struct and demonstrate field access.
6. Demonstrate how to use pointers with structs.
7. Write a function that accepts a struct by value and by pointer.
8. Modify a struct field inside a function.
9. Serialize a struct to JSON.
10. Deserialize JSON into a struct.

🔹 Interfaces (61–70)

1. Define a Shape interface with Area() method.
2. Implement the Shape interface for Circle and Rectangle.
3. Use type assertion to check underlying types.
4. Create a function that accepts any interface{} and prints type.
5. Demonstrate use of empty interface for generic values.
6. Show interface embedding (e.g., ReadWriter).
7. Create a custom interface and mock its implementation.
8. Compare two interface values.
9. Show how to implement an interface implicitly.
10. Write a program using polymorphism with interfaces.

🔹 Goroutines & Concurrency (71–85)

1. Launch a simple goroutine and observe its execution.
2. Use time.Sleep to allow goroutines to complete.
3. Launch multiple goroutines to print concurrently.
4. Use WaitGroup to wait for multiple goroutines.
5. Implement worker pool using goroutines.
6. Pass arguments safely to goroutines using closures.
7. Prevent race conditions with sync.Mutex.
8. Use sync.Once to ensure one-time execution.
9. Use atomic operations for safe counters.
10. Use select to multiplex channels.
11. Create a ticker using time.NewTicker.
12. Implement timeout using select and time.After.
13. Fan-out and fan-in pattern with channels.
14. Use context to cancel goroutines.
15. Limit concurrency using buffered channels.

🔹 Channels (86–95)

1. Create a channel and send/receive data.
2. Close a channel and check with comma-ok.
3. Use buffered channel and demonstrate blocking behavior.
4. Write a producer-consumer example with channels.
5. Pass channels as function parameters.
6. Use range to read from a channel until it’s closed.
7. Use channel of structs.
8. Implement a pipeline with multiple stages.
9. Use sync.Cond to coordinate goroutines.
10. Use channels to synchronize access to shared data.

🔹 Standard Library & Utilities (96–100)

1. Parse command-line arguments using flag package.
2. Read a file and print contents line by line.
3. Write data to a file in Go.
4. Perform HTTP GET request and read response.
5. Create a simple HTTP server with a handler.
6. Write a unit test for a function that adds two numbers.
7. Use table-driven tests for multiple input scenarios.
8. Test a function that returns an error.
9. Write a benchmark test using testing.B.
10. Use setup and teardown with testing.M.
11. Use go test with code coverage.
12. Create custom error types using errors.New.
13. Use fmt.Errorf with wrapping and formatting.
14. Use errors.Is and errors.As for error comparison.
15. Handle panics using defer and recover.
16. Demonstrate panic and recover with nested calls.
17. Write test cases for functions that panic.
18. Use testify/assert for more readable unit tests.
19. Mock dependencies using interfaces in tests.
20. Create integration tests for a small web API.

🔹 File I/O & OS Operations (116–125)

1. Read the contents of a text file.
2. Write data to a file (overwrite or append).
3. Check if a file or directory exists.
4. Read a file line-by-line using bufio.Scanner.
5. Copy the contents of one file to another.
6. Create and remove directories programmatically.
7. Get file info using os.Stat.
8. Walk through a directory tree and list files.
9. Rename or move files using os.Rename.
10. Change file permissions with os.Chmod.

🔹 Web & REST API (126–135)

1. Create a basic HTTP server with net/http.
2. Handle GET and POST requests in different endpoints.
3. Parse URL query parameters.
4. Parse JSON request body and respond with JSON.
5. Use http.ServeMux to define custom routers.
6. Implement middleware to log requests.
7. Return custom HTTP status codes and messages.
8. Handle path variables (e.g., /user/{id}) manually.
9. Serve static files (HTML, CSS, JS).
10. Create a simple REST API with CRUD operations.

🔹 Modules & Packages (136–145)

1. Initialize a Go module using go mod init.
2. Import and use a third-party package.
3. Create and use your own package across files.
4. Organize Go code into modules and packages.
5. Use replace directive in go.mod to override dependencies.
6. Create a reusable utility package.
7. Build and install a Go package locally.
8. Use go get to download and update packages.
9. Vendor dependencies and use go mod vendor.
10. Explain internal and exported identifiers in Go.

🔹 Advanced Concepts (146–150)

1. Explain memory escape analysis with example.
2. Demonstrate function currying using closures.
3. Use reflection to inspect struct fields at runtime.
4. Use unsafe package to manipulate memory (cautiously).
5. Implement dependency injection using interfaces and structs.

System Design in Go (151–165)

1. Design a URL shortener using Go.
2. Implement rate limiting for an API endpoint.
3. Design a key-value store using Go maps and channels.
4. Create an in-memory cache with expiration.
5. Build a message queue in Go using channels.
6. Design a logging system using goroutines and channels.
7. Implement a scheduler to run tasks at specific intervals.
8. Design a notification system (email/SMS) with worker pool.
9. Create a lightweight job dispatcher using goroutines.
10. Design a file upload server with progress tracking.
11. Build a user authentication system with JWT.
12. Design a distributed lock using etcd or Redis in Go.
13. Create a modular microservice with its own database.
14. Design a real-time chat server using WebSockets.
15. Implement pagination in an API.

🔹 Performance & Optimization (166–175)

1. Measure execution time of a function.
2. Profile memory and CPU usage of a Go program.
3. Identify and fix a memory leak.
4. Use pprof to monitor and diagnose performance issues.
5. Optimize slice usage to avoid unnecessary allocations.
6. Benchmark and compare two implementations of a function.
7. Avoid unnecessary goroutines in high-load systems.
8. Use sync.Pool for memory reuse.
9. Optimize JSON parsing in high-throughput applications.
10. Understand and avoid false sharing in concurrent code.

🔹 Networking & Low-Level (176–185)

1. Build a TCP server and client in Go.
2. Send and receive UDP packets.
3. Use bufio.Reader to handle partial TCP reads.
4. Implement a simple HTTP proxy server.
5. Parse a custom binary protocol.
6. Write a program that pings a server and prints RTT.
7. Create a raw socket in Go (with third-party library).
8. Use TLS to secure a client-server connection.
9. Resolve a domain name to an IP address using net.LookupIP.
10. Implement a DNS client using UDP.

🔹 gRPC & Protobuf (186–195)

1. Create a gRPC server and client in Go.
2. Define and compile a .proto file using protoc.
3. Implement unary RPC in Go.
4. Implement server-side streaming RPC in Go.
5. Implement client-side streaming RPC in Go.
6. Implement bidirectional streaming RPC in Go.
7. Add authentication to gRPC calls using metadata.
8. Use gRPC interceptors for logging and monitoring.
9. Generate Go code with gRPC-Gateway for REST support.
10. Use reflection and health checks in gRPC server.

🔹 Deployment & DevOps (196–200)

1. Build and run a Go project using go build and go run.
2. Create a Dockerfile for a Go application.
3. Build a multi-stage Docker image for minimal binary size.
4. Deploy a Go app to Kubernetes with a basic YAML spec.
5. Use environment variables and config files with Viper.

Cloud-Native & Kubernetes (201–215)

1. Create a Kubernetes Deployment for a Go application.
2. Expose a Go web app using a Kubernetes Service.
3. Implement health check endpoints (readiness & liveness).
4. Use Helm to deploy a templated Go service.
5. Build a gRPC service and deploy on Kubernetes.
6. Handle graceful shutdown in Go when receiving SIGTERM.
7. Configure and read environment variables in a containerized Go app.
8. Inject configuration using Kubernetes ConfigMap.
9. Inject secrets into a Go app using Kubernetes Secrets.
10. Autoscale a Go microservice using HPA (Horizontal Pod Autoscaler).
11. Use go-cloud to interact with cloud services like GCP/AWS.
12. Publish and consume messages with Google Pub/Sub in Go.
13. Access Amazon S3 from Go using AWS SDK.
14. Store logs in CloudWatch or Stackdriver from a Go service.
15. Instrument Go services with Prometheus metrics.

🔹 Security & Best Practices (216–225)

1. Sanitize user input in a Go web application.
2. Hash passwords using bcrypt in Go.
3. Implement JWT-based authentication in a REST API.
4. Protect against SQL injection in Go (with database/sql).
5. Serve HTTPS with a TLS certificate in Go.
6. Enable and validate CORS headers.
7. Perform rate limiting to prevent abuse.
8. Create and verify signed cookies.
9. Prevent directory traversal in file upload handlers.
10. Conduct input validation with go-playground/validator.

🔹 Advanced Concurrency Patterns (226–235)

1. Implement a throttler using channels and timers.
2. Write a concurrent map safely without sync.Map.
3. Build a reusable goroutine pool with a job queue.
4. Use errgroup.Group to manage multiple goroutines with shared error handling.
5. Implement a circuit breaker pattern in Go.
6. Apply the fan-in/fan-out pattern with multiple workers.
7. Detect and avoid goroutine leaks.
8. Use semaphore pattern to limit parallelism.
9. Combine context with channels for cancellation.
10. Implement a timeout mechanism using select and context.

🔹 Real-World Use Cases (236–245)

1. Create a CLI tool using Cobra.
2. Parse Excel or CSV files in Go.
3. Monitor a directory for changes using fsnotify.
4. Schedule periodic jobs using robfig/cron.
5. Export logs to ELK stack-compatible format.
6. Build a metrics exporter compatible with Prometheus.
7. Implement a reverse proxy with caching.
8. Create a PDF report from data using Go.
9. Upload files to AWS S3 and get public URLs.
10. Connect to PostgreSQL and run CRUD queries.

🔹 Design & Architecture (246–250)

1. Explain hexagonal (ports and adapters) architecture in Go.
2. Use domain-driven design (DDD) principles in a Go project.
3. Separate interfaces from implementations for better testing.
4. Build a scalable logging framework using interfaces.
5. Use dependency injection to write loosely coupled code.

Database Access & Optimization (251–265)

1. Connect to a PostgreSQL database using database/sql.
2. Use prepared statements to prevent SQL injection.
3. Use context for query timeout/cancellation.
4. Create transactions and handle commit/rollback.
5. Use SQL joins to fetch related data.
6. Scan query results into a custom struct.
7. Batch insert records into a database.
8. Use GORM to perform CRUD operations.
9. Add hooks in GORM for auditing fields.
10. Use database indexes to improve performance (explain how).
11. Optimize slow queries using EXPLAIN.
12. Implement soft deletes in GORM.
13. Use database migrations with goose or sql-migrate.
14. Configure connection pooling for a DB in Go.
15. Handle DB failover or retries using exponential backoff.

🔹 Code Generation & Meta Programming (266–275)

1. Generate Go code from .proto files using protoc-gen-go.
2. Use go generate for automatic code generation.
3. Generate mock interfaces using mockgen.
4. Create a code generator using templates and AST.
5. Use reflection to inspect struct field tags.
6. Implement custom JSON marshalling logic.
7. Dynamically invoke a function using reflect.
8. Use go:generate to embed assets into binaries.
9. Understand and use struct tags effectively.
10. Create custom template functions in text/template.

🔹 Caching & Performance Engineering (276–285)

1. Implement in-memory LRU cache.
2. Integrate Redis for caching data in Go.
3. Cache API responses with TTL.
4. Use go-cache for expiring in-memory data.
5. Add cache invalidation logic based on events.
6. Benchmark cached vs. non-cached function calls.
7. Use a cache-aside pattern in Go.
8. Monitor cache hit/miss ratio.
9. Prevent thundering herd with singleflight.
10. Cache static configuration in memory.

🔹 Observability: Logs, Traces, Metrics (286–295)

1. Log structured data using logrus or zap.
2. Implement middleware for request logging.
3. Track request latency using Prometheus metrics.
4. Add tracing with OpenTelemetry in Go.
5. Trace HTTP and gRPC requests end-to-end.
6. Export metrics to Prometheus using /metrics endpoint.
7. Log and export panics for alerting.
8. Use Grafana to visualize service metrics.
9. Configure log rotation using lumberjack.
10. Implement distributed tracing with Jaeger and Go.

🔹 Event-Driven Architecture (296–300)

1. Publish/subscribe to events using NATS or RabbitMQ.
2. Handle incoming Kafka events in Go using sarama.
3. Create a publisher/subscriber interface for abstraction.
4. Implement an event dispatcher pattern.
5. Use domain events to decouple service layers.
6. Implement the Singleton pattern in Go.
7. Implement the Factory pattern with interfaces.
8. Use the Strategy pattern to switch algorithms at runtime.
9. Demonstrate the Decorator pattern with I/O.
10. Implement the Observer pattern for an event system.
11. Use the Command pattern to queue and execute tasks.
12. Demonstrate the Adapter pattern with a third-party lib.
13. Implement the Proxy pattern for caching/mocking.
14. Create a Chain of Responsibility for middleware chaining.
15. Use the Template Method pattern for extensible behavior.
16. Apply the Builder pattern to build complex structs.
17. Implement the State pattern for a finite state machine.
18. Apply the Memento pattern to undo/redo state changes.
19. Design a Pub/Sub interface and its implementations.
20. Use the Iterator pattern to traverse custom collections.

🔹 Go Internals & Compiler (316–325)

1. Explain how Go’s garbage collector works.
2. What is escape analysis in Go? Show with example.
3. Compare stack vs. heap allocation in Go.
4. What are goroutine scheduling strategies?
5. Understand Go’s memory layout of structs.
6. How does Go's import cycle detection work?
7. How are maps implemented in Go internally?
8. What happens during go build? Step-by-step.
9. Understand defer statement internals and performance.
10. Understand how interface values are represented in memory.

🔹 Build Tools & Automation (326–335)

1. Create a Makefile to build, test, and run a Go app.
2. Use go fmt and go vet in a CI pipeline.
3. Set up a GitHub Actions workflow for testing Go code.
4. Use pre-commit hooks to lint and format Go files.
5. Build and tag Go binaries with version info.
6. Cross-compile a Go app for multiple OS/architectures.
7. Integrate Go tests with a coverage report in CI.
8. Build Docker images on push using CI/CD tools.
9. Use Goreleaser to publish binaries and Docker images.
10. Use go install to distribute CLI tools.

🔹 Distributed Systems in Go (336–345)

1. Build a consistent hash ring implementation.
2. Use Raft consensus algorithm (via Hashicorp Raft).
3. Design and build a simple service discovery mechanism.
4. Implement leader election with distributed coordination.
5. Simulate network partitions and recoveries.
6. Use gRPC with service reflection and health checks.
7. Use circuit breakers to isolate service failures.
8. Add retries with exponential backoff in RPC clients.
9. Build a distributed key-value store prototype.
10. Handle service timeouts and fallbacks gracefully.

🔹 Real-World Software Architecture (346–350)

1. Apply clean architecture principles to a Go service.
2. Build a multi-tenant REST API in Go.
3. Design a horizontally scalable file processing pipeline.
4. Separate domain and infrastructure concerns effectively.
5. Build a plug-in system using interfaces and reflection.

Advanced Go & Systems Programming (351–365)

1. Use cgo to call a C function from Go.
2. Call Go functions from C using export and cgo.
3. Manage memory safely when using cgo.
4. Use Go assembly for performance-critical code.
5. Benchmark a cgo vs pure-Go function.
6. Monitor and limit memory usage in a Go program.
7. Use syscall for low-level system interaction.
8. Read from /proc or use os packages to inspect processes.
9. Pin a goroutine to a thread with runtime.LockOSThread.
10. Use go:linkname to access unexported symbols (advanced).
11. Use Go to parse and interpret custom binary protocols.
12. Capture and analyze stack traces in production.
13. Implement file locking across processes.
14. Use Go to communicate with a device via serial port.
15. Use mmap in Go to map a file into memory.

🔹 AI, ML & Data Engineering with Go (366–375)

1. Read and preprocess a CSV dataset in Go.
2. Use Gorgonia for matrix operations and neural nets.
3. Build a simple linear regression model in Go.
4. Implement a K-means clustering algorithm.
5. Serialize a trained model and use it in Go.
6. Use ONNX Go bindings to load ML models.
7. Read and write Parquet files in Go.
8. Stream process large datasets using goroutines.
9. Build an ETL pipeline in Go for batch jobs.
10. Use Go to interact with Apache Kafka for real-time ingestion.

🔹 CLI Development & Tooling (376–385)

1. Build a CLI app using Cobra.
2. Use urfave/cli to create subcommands and flags.
3. Create interactive prompts using survey package.
4. Parse config files with viper and bind flags/envs.
5. Use spf13/cast to safely cast interface{} to types.
6. Build a text-based UI using tview.
7. Write unit and integration tests for CLI commands.
8. Package and install a CLI app with go install.
9. Generate shell completions for a CLI tool.
10. Parse JSON/YAML input passed via stdin.

🔹 Cross-Language & Integration (386–390)

1. Use gRPC to connect Go and Python microservices.
2. Call JavaScript from Go using syscall/js (WASM).
3. Call Python code from Go via shared library.
4. Use Go plugins for dynamic module loading.
5. Embed a Lua or Python interpreter in a Go program.

🔹 Large-Scale & Production Systems (391–400)

1. Build a scalable task queue using Redis and Go workers.
2. Implement request deduplication in distributed systems.
3. Handle eventual consistency in Go-based services.
4. Manage hundreds of concurrent DB queries safely.
5. Process 1M+ records in parallel with backpressure.
6. Use gRPC + Protobuf with versioning strategies.
7. Gracefully restart a Go service with zero downtime.
8. Use structured logging across distributed services.
9. Implement API gateway in Go with routing + auth.
10. Debug memory leaks and deadlocks in a live Go system.
11. Build a GraphQL API using gqlgen in Go.
12. Define types and resolvers in a gqlgen schema.
13. Handle GraphQL queries and mutations with Go structs.
14. Use context in resolvers for authentication.
15. Implement query batching and caching in GraphQL.
16. Add custom directives and scalars in gqlgen.
17. Combine REST and GraphQL endpoints in a single Go server.
18. Implement pagination in GraphQL (Relay-style or offset).
19. Secure a GraphQL API with JWT and middleware.
20. Use dataloaders to prevent N+1 query problems.

🔹 Go in Serverless & Cloud Functions (411–420)

1. Deploy a Go function to AWS Lambda.
2. Write a Go function for Google Cloud Functions.
3. Deploy Azure Functions using Go and VSCode CLI.
4. Use API Gateway to trigger Go Lambda with JSON input.
5. Optimize cold starts for Go in serverless platforms.
6. Handle large file uploads in serverless Go apps.
7. Use Terraform to provision Go-based Lambdas.
8. Handle concurrency and timeouts in serverless Go.
9. Monitor Go functions with X-Ray or Cloud Trace.
10. Use Go to create serverless event processors (e.g. S3 triggers).

🔹 Go & Service Mesh / Cloud-Native (421–430)

1. Instrument a Go app with OpenTelemetry for tracing.
2. Expose custom metrics in Prometheus format from a Go service.
3. Use Linkerd or Istio with a Go microservice.
4. Understand mTLS and implement it in a Go HTTP server.
5. Configure circuit breaking and retries via service mesh.
6. Use Go to propagate headers through a mesh (e.g. trace ID).
7. Set up a Go app behind an ingress gateway.
8. Implement client-side load balancing in Go.
9. Use gRPC Load Balancer policies with Go clients.
10. Build a sidecar in Go that logs and routes traffic.

🔹 Go for Embedded & IoT Development (431–440)

1. Cross-compile Go code for Raspberry Pi.
2. Read GPIO pins using periph.io in Go.
3. Send/receive data over UART (serial) in Go.
4. Interface with sensors over I2C or SPI using Go.
5. Use TinyGo to compile Go for microcontrollers.
6. Flash a Go program to an ESP32 using TinyGo.
7. Capture images from camera and process in Go.
8. Send sensor data to MQTT broker using Go.
9. Build a Go service to aggregate telemetry from devices.
10. Secure communication between IoT device and server (TLS + JWT).

🔹 Games, Graphics & UI in Go (441–450)

1. Create a simple 2D game using Ebiten.
2. Handle keyboard and mouse input in Go.
3. Load and render a sprite sheet in Ebiten.
4. Use raylib-go to build a 3D environment.
5. Animate a player character with frame control.
6. Track game score and state using structs.
7. Detect collisions between objects in Go.
8. Export game build for desktop and web (WASM).
9. Integrate sound effects and background music.
10. Build a level editor using Gio or Fyne GUI framework.

Machine Learning & Go APIs (451–460)

1. Build a RESTful prediction API with a pre-trained model in Go.
2. Call a Python ML service from Go using gRPC or HTTP.
3. Use TensorFlow Go bindings to run inference.
4. Serialize Go data structures to JSON for ML APIs.
5. Integrate Go with HuggingFace inference API.
6. Visualize prediction results using Go and a plotting lib.
7. Stream real-time predictions using websockets.
8. Optimize ML API latency using goroutines.
9. Build a batch scoring pipeline in Go.
10. Create a feedback loop system for online learning (in Go).

🔹 WebAssembly (WASM) with Go (461–470)

1. Compile a Go program to WebAssembly.
2. Use syscall/js to manipulate DOM from Go.
3. Expose Go functions to JavaScript via WASM.
4. Create a browser-based calculator using Go + WASM.
5. Handle events (click, input) in Go WASM apps.
6. Build a WASM module that runs image filters in-browser.
7. Optimize Go WASM bundle size.
8. Communicate between Go and JS using JSON.
9. Deploy Go WASM app to static site hosting.
10. Use TinyGo to compile for lightweight WASM.

🔹 Blockchain & Web3 in Go (471–480)

1. Connect to Ethereum using go-ethereum (geth).
2. Create a wallet and sign transactions in Go.
3. Call a smart contract function using Go.
4. Listen to blockchain events using Go.
5. Create a simple blockchain prototype in Go.
6. Use Go to parse and index blocks.
7. Build a REST API for querying smart contract state.
8. Write a CLI for managing Ethereum keys.
9. Use IPFS in Go to store and retrieve files.
10. Securely store private keys using Go crypto libs.

🔹 Advanced Testing & Quality (481–490)

1. Fuzz test a function using go test -fuzz.
2. Detect race conditions using -race flag.
3. Use t.TempDir for temporary test directories.
4. Write table-driven tests for HTTP handlers.
5. Benchmark parallel execution of goroutines.
6. Use fake time in tests with clock packages.
7. Capture logs during tests using test writers.
8. Run tests in CI with code coverage report.
9. Generate test data using faker libraries.
10. Snapshot test JSON API responses.

🔹 Logging & Distributed Tracing Pipelines (491–500)

1. Set up centralized logging in Go with Logstash/Fluentbit.
2. Format logs as JSON for shipping to ELK/Datadog.
3. Correlate logs with trace/span IDs in Go apps.
4. Export distributed traces with OpenTelemetry.
5. Instrument database queries with tracing.
6. Visualize tracing data in Jaeger/Grafana Tempo.
7. Create a tracing middleware for gRPC.
8. Send application metrics to a remote Prometheus PushGateway.
9. Build an observability sidecar in Go.
10. Use structured logging to capture errors with full context.
11. Build a Go backend for a Flutter or React Native mobile app
12. Use Go with Electron to build a desktop GUI
13. Call Go functions from Rust via shared libraries
14. Use gRPC to bridge Go services with Node.js or Java
15. Convert Go structs into GraphQL schemas dynamically
16. Sync data between Go and a Firebase backend
17. Interact with SQLite on mobile via Go Mobile Bindings
18. Use Go to power backend APIs for a PWA
19. Bundle Go + WASM with a React frontend
20. Connect a Go microservice to a Python FastAPI via Protobuf

🔹 Build Systems & Dev Efficiency (511–520)

1. Use mage as a modern alternative to Makefile in Go
2. Automate dependency upgrades in Go with Renovate or Dependabot
3. Use build tags to control cross-platform builds
4. Create a monorepo with multiple Go modules
5. Use Bazel to build and test a large Go project
6. Minimize binary size using linker flags (e.g. -s -w)
7. Set up caching in CI/CD to speed up go mod downloads
8. Build a custom static code analyzer for Go (using go/ast)
9. Use GoReleaser to manage versioning and changelogs
10. Benchmark and monitor build time regressions

🔹 Time Series & Real-Time Processing (521–530)

1. Parse and aggregate logs by timestamp in Go
2. Build a real-time dashboard with WebSockets and Go
3. Write a simple time-series database (TSDB) in Go
4. Store time series in InfluxDB from a Go client
5. Implement exponential decay counters in Go
6. Stream data from Kafka and batch-write to a DB
7. Handle out-of-order time series events
8. Build a heatmap with Go and a frontend chart library
9. Detect anomalies in time series with statistical functions
10. Use go-chrono or time.Time for timezone-aware math

🔹 Functional Programming Concepts in Go (531–540)

1. Create a generic map/reduce function using Go 1.18+ generics
2. Implement a filter function for a slice of structs
3. Write and compose higher-order functions
4. Use closures to capture loop variables safely
5. Create a lazy sequence generator with channels
6. Handle immutability using copy-on-write patterns
7. Use option types or monads in Go idiomatically
8. Simulate pattern matching using type switches
9. Combine multiple operations with function pipelines
10. Apply the Either or Result pattern to error handling

🔹 Real-World Engineering Scenarios (541–550)

1. Design a rate-limiting library usable across services
2. Migrate from monolith to microservices using Go
3. Analyze and refactor a legacy Go codebase
4. Run blue-green deployments for Go microservices
5. Build a disaster recovery plan for a Go system
6. Write a Go client for a third-party API (e.g., Stripe)
7. Handle mobile push notifications from a Go backend
8. Design a Go-based config service (feature flags, toggles)
9. Build a self-hosted file upload API with virus scanning
10. Create an internal developer platform CLI in Go

Edge Computing & IoT (551–560)

1. Deploy a Go app to edge devices with Kubernetes (K3s).
2. Implement local caching for offline-first edge apps.
3. Use MQTT in Go for lightweight messaging at the edge.
4. Secure edge devices using mutual TLS in Go.
5. Monitor and update firmware over-the-air with Go services.
6. Build an edge aggregator to batch and forward data.
7. Use WebRTC with Go for peer-to-peer device communication.
8. Implement time synchronization between edge nodes.
9. Use gRPC-Web for browser-edge communication.
10. Build a resource-constrained Go service for microcontrollers.

🔹 Streaming Protocols & Data Pipelines (561–570)

1. Build a Kafka producer and consumer with sarama.
2. Use Apache Pulsar client in Go for topic subscription.
3. Create a real-time data enrichment pipeline in Go.
4. Handle backpressure when consuming high-volume streams.
5. Write a Go app that processes WebSocket streams.
6. Use Apache Flink or Beam with Go connectors.
7. Implement a sliding window aggregation over streaming data.
8. Serialize data efficiently with Cap’n Proto or FlatBuffers.
9. Build a custom streaming protocol over TCP in Go.
10. Use context cancellation to gracefully stop stream processing.

🔹 Peer-to-Peer Networking & Decentralization (571–580)

1. Build a simple P2P network with Go using libp2p.
2. Implement peer discovery and NAT traversal in Go.
3. Build a decentralized chat app with P2P connections.
4. Use DHT (Distributed Hash Table) in Go for lookup.
5. Create a secure messaging protocol with end-to-end encryption.
6. Handle network partitions and peer reconnection logic.
7. Design a gossip protocol for cluster state sync.
8. Build a file sharing P2P app with chunking and verification.
9. Use IPFS APIs in Go for decentralized storage.
10. Benchmark P2P network latency and throughput.

🔹 Go for Hardware & Systems Control (581–590)

1. Control GPIO pins on embedded Linux using Go.
2. Use libusb or hidapi in Go to communicate with USB devices.
3. Read and write to SPI flash memory using Go.
4. Implement a simple device driver in Go (via cgo or userspace).
5. Build a serial port terminal in Go for device debugging.
6. Use CAN bus protocol libraries in Go.
7. Interface with Bluetooth Low Energy devices in Go.
8. Monitor hardware sensors (temperature, voltage) with Go.
9. Write a watchdog timer service to restart stuck processes.
10. Build a Go service that controls robotics actuators.

🔹 System Reliability & Observability (591–600)

1. Implement chaos engineering tests in Go to simulate failures.
2. Build automated health checks and self-healing routines.
3. Use eBPF tooling with Go for kernel-level monitoring.
4. Design and build a distributed tracing correlation ID generator.
5. Implement log sampling to reduce volume in high-throughput systems.
6. Detect and alert on abnormal system metrics programmatically.
7. Build a circuit breaker middleware with fallback support.
8. Implement rate-limiting middleware with token bucket algorithm.
9. Design graceful shutdown with draining connections in Go servers.
10. Use Go runtime metrics for proactive scaling decisions.

Data Science & Analytics in Go (601–610)

1. Parse large JSON datasets efficiently with streaming decoders.
2. Perform matrix multiplications using Gonum library.
3. Implement principal component analysis (PCA) from scratch.
4. Build a time series forecasting model with Go.
5. Use Go bindings to Apache Arrow for in-memory analytics.
6. Compute statistical summaries (mean, median, stddev) on large datasets.
7. Write a Go program to visualize data using plotting libs like gonum/plot.
8. Perform linear algebra operations with generic types in Go 1.18+.
9. Use Go routines and channels for parallel data processing.
10. Build a recommender system using collaborative filtering in Go.

🔹 Advanced Blockchain & Smart Contracts (611–620)

1. Write Go code to interact with NFT smart contracts.
2. Implement token transfer functions using go-ethereum.
3. Decode and encode Ethereum transaction data.
4. Monitor mempool transactions in real-time with Go.
5. Build a blockchain explorer backend in Go.
6. Write Go tests for smart contract event parsing.
7. Implement zero-knowledge proof verification in Go.
8. Use Go to interact with Layer 2 solutions like Polygon.
9. Build a cross-chain bridge prototype in Go.
10. Securely manage keys using hardware wallets in Go.

🔹 Performance Profiling & Optimization (621–630)

1. Profile CPU and memory usage with pprof in Go.
2. Analyze goroutine blocking and deadlocks.
3. Use go tool trace to visualize execution.
4. Optimize hot paths with assembly or intrinsics.
5. Reduce GC pauses by tuning object allocations.
6. Measure latency distributions with histogram libraries.
7. Benchmark IO operations using testing.B.
8. Detect memory leaks with heap profiles.
9. Profile network IO and optimize connection pooling.
10. Use runtime/metrics for custom metrics collection.

🔹 Cloud-Native & Kubernetes Go Tools (631–640)

1. Build a Kubernetes operator using controller-runtime.
2. Use client-go to interact with Kubernetes API.
3. Write admission controllers in Go.
4. Use Go for writing custom resource definitions (CRDs).
5. Automate deployments with Helm charts and Go templates.
6. Use Go to build a webhook server for Kubernetes.
7. Create custom metrics for Prometheus scraping in Go.
8. Build a Kubernetes scheduler extender in Go.
9. Implement health probes (readiness, liveness) in Go.
10. Use Go to watch and reconcile Kubernetes resources.

🔹 Security & Cryptography in Go (641–650)

1. Implement AES-GCM encryption and decryption in Go.
2. Use Go crypto/tls for secure client-server communication.
3. Generate and validate JWT tokens securely.
4. Use Go to create a secure password hashing function (bcrypt/scrypt).
5. Implement HMAC-based request signing for APIs.
6. Use x509 certificates for mutual TLS in Go.
7. Protect against timing attacks in crypto code.
8. Use libsodium bindings for advanced cryptography.
9. Build a secure key-value store with encrypted data at rest.
10. Audit Go code for common security vulnerabilities.
11. Implement a custom HTTP/2 server in Go.
12. Parse and build DNS packets using Go libraries.
13. Build a SOCKS5 proxy server with Go.
14. Implement QUIC protocol basics in Go.
15. Write a TCP load balancer with connection multiplexing.
16. Build a secure WebSocket server with authentication.
17. Implement an MQTT broker in Go.
18. Use SCTP protocol support in Go (via external libs).
19. Build a network packet sniffer with gopacket.
20. Create a NAT traversal service using ICE and STUN.

🔹 Concurrency & Parallelism Patterns (661–670)

1. Implement a worker pool with graceful shutdown.
2. Design a concurrent cache with read-write locks.
3. Use atomic operations for lock-free counters.
4. Build a rate limiter using Go channels and timers.
5. Implement a parallel map function using goroutines.
6. Use errgroup for managing concurrent task groups.
7. Build a fan-in/fan-out pipeline for data processing.
8. Handle cancellation and timeouts with contexts.
9. Create a concurrent-safe linked list or queue.
10. Implement a publish-subscribe pattern with Go channels.

🔹 Go for Big Data & ETL (671–680)

1. Parse and transform large CSV files efficiently.
2. Use Apache Spark or Hadoop connectors with Go.
3. Write Go code to interact with BigQuery.
4. Build a data ingestion pipeline with Apache Kafka.
5. Process Parquet files using Go libraries.
6. Use Go routines to parallelize ETL jobs.
7. Build a metadata catalog in Go for datasets.
8. Implement incremental data sync in Go.
9. Handle schema evolution during data ingestion.
10. Automate data quality checks with Go tests.

🔹 Cloud Automation & Infrastructure as Code (681–690)

1. Use Go with Terraform SDK to write custom providers.
2. Automate AWS resource provisioning with Go AWS SDK.
3. Manage Azure resources via Go SDK.
4. Build a CLI tool in Go to manage Kubernetes clusters.
5. Write a GitOps operator in Go for config management.
6. Use Go to automate Docker image builds and pushes.
7. Create an auto-scaling controller using Go and Kubernetes API.
8. Use Go to monitor cloud billing and usage data.
9. Build multi-cloud deployment tools in Go.
10. Implement secret management integration (Vault, AWS KMS) in Go.

🔹 Testing & Debugging at Scale (691–700)

1. Implement chaos testing frameworks for Go services.
2. Build custom test mocks for external dependencies.
3. Use property-based testing with gopter.
4. Profile Go applications in production with minimal overhead.
5. Implement canary testing pipelines using Go tools.
6. Automate rollback on failed deployments with Go scripts.
7. Use distributed tracing to debug cross-service issues.
8. Write fuzz tests targeting security-critical code.
9. Analyze goroutine dump files to identify deadlocks.
10. Build dashboards for real-time test results visualization.

**Golang Interview Coding Questions: Set 15 (701–750)**

**701.** Write a function to reverse a string in Go.  
**702.** How do you implement a binary search tree (BST) in Go?  
**703.** Explain the difference between slice length and capacity.  
**704.** Implement a function to check if a linked list has a cycle.  
**705.** How do you create and manage context with deadlines and cancellation?  
**706.** Write a Go function to merge two sorted slices.  
**707.** Explain the difference between pointer receivers and value receivers in methods.  
**708.** How do you implement a semaphore using channels?  
**709.** Write a function to flatten a nested slice of integers.  
**710.** How does Go handle package initialization order?  
**711.** Write a function to check if a string is a palindrome.  
**712.** Explain how sync.Mutex and sync.RWMutex differ.  
**713.** Implement a concurrent-safe LRU cache in Go.  
**714.** How do you implement interface assertion and type switches?  
**715.** Write a function to perform a depth-first search (DFS) on a graph.  
**716.** Explain the use of unsafe package and when it’s appropriate.  
**717.** How do you limit the number of goroutines running concurrently?  
**718.** Write a Go program to read and write CSV files.  
**719.** Explain how to use the select statement with timeout cases.  
**720.** Write a function to calculate factorial using recursion and iteration.  
**721.** How do you implement error wrapping and unwrapping?  
**722.** Write a function to implement a binary heap (min-heap or max-heap).  
**723.** Explain the use of go generate and how it helps in code generation.  
**724.** How do you implement a publish-subscribe pattern without external libraries?  
**725.** Write a program to perform concurrent HTTP requests and gather results.  
**726.** Explain the difference between buffered and unbuffered channels with examples.  
**727.** How do you handle JSON with dynamic or unknown fields?  
**728.** Write a Go function to convert a string to title case.  
**729.** Explain how method sets affect interface implementation.  
**730.** Write a function to detect duplicate elements in a slice.  
**731.** How do you implement retries with exponential backoff?  
**732.** Explain the difference between embedding and composition in Go.  
**733.** Write a function to find the longest substring without repeating characters.  
**734.** How do you use reflection to access struct fields dynamically?  
**735.** Implement a concurrent pipeline pattern using goroutines and channels.  
**736.** Explain how Go modules work and how to manage dependencies.  
**737.** Write a function to serialize and deserialize data using Gob encoding.  
**738.** How do you handle panics and recover gracefully?  
**739.** Write a function to find the intersection of two slices.  
**740.** Explain the use and benefits of context in HTTP servers.  
**741.** Write a Go program to implement a simple TCP server and client.  
**742.** How do you perform unit testing for functions that use time?  
**743.** Implement a balanced binary search tree (AVL or Red-Black Tree) in Go.  
**744.** Explain how Go handles memory alignment and struct padding.  
**745.** Write a function to convert a Roman numeral to an integer.  
**746.** How do you implement a thread-safe singleton pattern?  
**747.** Write a Go function to paginate a slice of data.  
**748.** Explain how to optimize Go code for reduced memory allocations.  
**749.** Write a function to check if two strings are anagrams.  
**750.** Explain the pros and cons of using cgo in Go projects.

**Golang Interview Coding Questions: Set 16 (751–800)**

**751.** Write a function to rotate an array by k positions.  
**752.** How do you implement a priority queue in Go?  
**753.** Explain how to use the context package to propagate cancellation across API calls.  
**754.** Write a Go program to parse command-line flags using the flag package.  
**755.** How do you implement a rate limiter with a token bucket algorithm?  
**756.** Write a function to find the first non-repeating character in a string.  
**757.** Explain the difference between panic, recover, and defer.  
**758.** Write a function to merge intervals in a list of intervals.  
**759.** How do you implement a thread-safe counter?  
**760.** Write a Go function to perform breadth-first search (BFS) on a graph.  
**761.** Explain how to use interfaces for mocking in unit tests.  
**762.** Write a function to find the maximum subarray sum (Kadane’s algorithm).  
**763.** How do you handle SQL database transactions in Go?  
**764.** Write a function to convert a slice of strings to uppercase concurrently.  
**765.** Explain the difference between sync.WaitGroup and sync.Cond.  
**766.** Write a Go function to detect if two strings are rotations of each other.  
**767.** How do you parse and validate URLs in Go?  
**768.** Write a program to implement a circular buffer.  
**769.** Explain the use of the init() function in Go packages.  
**770.** Write a function to compute the nth Fibonacci number using memoization.  
**771.** How do you implement a worker pool that processes jobs concurrently?  
**772.** Write a Go program to read a file line by line efficiently.  
**773.** Explain how Go handles slices internally (slice header).  
**774.** Write a function to check if a binary tree is a valid binary search tree.  
**775.** How do you implement logging with different levels (info, warn, error) in Go?  
**776.** Write a function to calculate the GCD (Greatest Common Divisor) of two numbers.  
**777.** Explain the difference between embedding a struct and embedding an interface.  
**778.** Write a Go function to convert an integer to its binary representation as a string.  
**779.** How do you implement exponential backoff with jitter in retries?  
**780.** Write a program to compress and decompress data using gzip in Go.  
**781.** Explain the difference between defer in loops versus outside loops.  
**782.** Write a function to check if a string contains only digits.  
**783.** How do you create custom errors with additional context?  
**784.** Write a function to find all permutations of a string.  
**785.** Explain how Go’s select statement handles multiple ready channels.  
**786.** Write a program to implement a basic pub/sub system with channels.  
**787.** How do you test private functions in Go?  
**788.** Write a function to count the frequency of words in a text file.  
**789.** Explain the concept of zero value and how it helps prevent bugs.  
**790.** Write a Go function to convert a map to a JSON string.  
**791.** How do you limit the size of a file upload in an HTTP server?  
**792.** Write a function to implement quicksort on a slice of integers.  
**793.** Explain how Go handles method receivers with embedded structs.  
**794.** Write a program to implement a countdown timer using goroutines and channels.  
**795.** How do you implement a simple HTTP middleware in Go?  
**796.** Write a function to reverse the words in a sentence.  
**797.** Explain how to use Go’s reflect package to dynamically call methods.  
**798.** Write a function to check if a linked list is a palindrome.  
**799.** How do you handle graceful shutdown in a Go HTTP server?  
**800.** Write a function to convert CSV data to a slice of structs.

**Golang Interview Coding Questions: Set 17 (801–850)**

**801.** Write a function to merge two sorted linked lists.  
**802.** How do you implement a Bloom filter in Go?  
**803.** Explain the difference between value and pointer semantics in Go maps.  
**804.** Write a program to implement a debounce function using Go channels.  
**805.** How do you use sync.Pool for object reuse?  
**806.** Write a function to find the median of two sorted arrays.  
**807.** Explain how Go’s escape analysis affects memory allocation.  
**808.** Write a program to perform concurrent file downloads with progress reporting.  
**809.** How do you handle JSON unmarshaling into interface types?  
**810.** Write a function to generate permutations of an integer slice.  
**811.** Explain the use of sync.Once and provide an example.  
**812.** Write a function to detect if two linked lists intersect.  
**813.** How do you implement deadlock detection in Go programs?  
**814.** Write a Go program to implement a Trie data structure.  
**815.** Explain how to use Go’s embedding to simulate inheritance.  
**816.** Write a function to serialize and deserialize a binary tree.  
**817.** How do you implement a read-write lock using sync.RWMutex?  
**818.** Write a program to detect and remove duplicates from a linked list.  
**819.** Explain how you can profile a Go application to find bottlenecks.  
**820.** Write a function to calculate the power of a number using recursion.  
**821.** How do you implement an event dispatcher in Go?  
**822.** Write a Go program to parse and manipulate XML data.  
**823.** Explain the role of build tags and how to use them.  
**824.** Write a function to find the kth largest element in a slice.  
**825.** How do you implement a generic stack data structure in Go using generics?  
**826.** Write a program to perform depth-first search on a maze.  
**827.** Explain how Go handles interface nil vs typed nil values.  
**828.** Write a function to find all anagrams of a string within another string.  
**829.** How do you implement a configurable logger with different output targets?  
**830.** Write a Go function to safely update a map concurrently.  
**831.** Explain how you can create custom JSON marshaling behavior.  
**832.** Write a program to detect cycles in a directed graph.  
**833.** How do you use channels to implement a barrier synchronization?  
**834.** Write a function to compute the edit distance between two strings.  
**835.** Explain the use of Go’s embed package to include static assets.  
**836.** Write a program to implement a circular linked list.  
**837.** How do you use reflection to create a deep copy of a struct?  
**838.** Write a function to convert a string containing a number in any base to an integer.  
**839.** Explain how to implement middleware chaining in an HTTP server.  
**840.** Write a program to simulate a rate limiter using Go’s time.Ticker.  
**841.** How do you handle versioning in Go modules?  
**842.** Write a function to find the maximum depth of a binary tree.  
**843.** Explain how to safely share data between goroutines.  
**844.** Write a Go program to implement a min stack (stack with minimum retrieval in O(1)).  
**845.** How do you perform unit testing for functions that use randomness?  
**846.** Write a function to convert a linked list to a slice.  
**847.** Explain how Go’s escape analysis helps optimize heap allocations.  
**848.** Write a program to implement a basic key-value store in memory.  
**849.** How do you parse command-line arguments with subcommands in Go?  
**850.** Write a function to serialize and deserialize JSON with custom field names.

**Golang Interview Coding Questions: Set 18 (851–900)**

**851.** Write a function to rotate a linked list to the right by k places.  
**852.** How do you implement a timeout for a goroutine?  
**853.** Explain the difference between method expressions and method values in Go.  
**854.** Write a Go program to implement a bloom filter with adjustable false positive rate.  
**855.** How do you perform dependency injection in Go?  
**856.** Write a function to validate an IPv4 address string.  
**857.** Explain how Go manages stack growth for goroutines.  
**858.** Write a program to generate Fibonacci numbers using channels.  
**859.** How do you implement a thread-safe queue in Go?  
**860.** Write a function to find the lowest common ancestor (LCA) in a binary tree.  
**861.** Explain the use of sync.Map and when to use it over regular maps.  
**862.** Write a function to implement a circular queue.  
**863.** How do you write benchmarks in Go?  
**864.** Write a Go program to parse and generate YAML data.  
**865.** Explain how Go’s compiler handles inlining of functions.  
**866.** Write a function to merge k sorted linked lists.  
**867.** How do you implement a memory pool to reduce garbage collection overhead?  
**868.** Write a function to find all pairs in an array that sum to a target value.  
**869.** Explain how Go’s interface values store type and value information.  
**870.** Write a Go program to implement a simple chat server using WebSocket.  
**871.** How do you implement exponential backoff with jitter?  
**872.** Write a function to find the longest common prefix in a slice of strings.  
**873.** Explain the difference between runtime.GOMAXPROCS and goroutine scheduling.  
**874.** Write a Go function to convert a slice of bytes to a hex string.  
**875.** How do you implement graceful restart in a Go HTTP server?  
**876.** Write a program to implement a simple load balancer using Go.  
**877.** Explain how to use the testing package to mock dependencies.  
**878.** Write a function to flatten a multi-dimensional slice recursively.  
**879.** How do you handle circular dependencies in Go packages?  
**880.** Write a Go function to detect if a number is prime.  
**881.** Explain the difference between Go’s value semantics and pointer semantics with examples.  
**882.** Write a function to calculate the sum of digits of a number recursively.  
**883.** How do you profile Go programs for CPU and memory usage?  
**884.** Write a program to implement a binary search algorithm on a sorted slice.  
**885.** Explain the role of go mod tidy and go mod vendor.  
**886.** Write a function to calculate the Hamming distance between two strings.  
**887.** How do you implement a generic queue using Go generics?  
**888.** Write a program to read JSON data from an HTTP request and respond accordingly.  
**889.** Explain the difference between a nil interface and an interface holding a nil pointer.  
**890.** Write a function to convert a number from decimal to hexadecimal string.  
**891.** How do you implement a timeout for an HTTP client request in Go?  
**892.** Write a Go program to implement a simple key-value store with expiration.  
**893.** Explain how to manage concurrency with worker pools and job queues.  
**894.** Write a function to find the smallest missing positive integer in an array.  
**895.** How do you implement an LRU cache with expiration times?  
**896.** Write a Go program to monitor file changes using fsnotify.  
**897.** Explain the difference between type embedding and interface embedding.  
**898.** Write a function to calculate the sum of elements in a slice using goroutines.  
**899.** How do you debug race conditions in Go programs?  
**900.** Write a Go program to parse and execute templates with dynamic data.

**Golang Interview Coding Questions: Set 19 (901–950)**

**901.** Write a function to find the longest palindrome substring in a string.  
**902.** How do you implement a thread-safe publish-subscribe system?  
**903.** Explain how to use Go’s context for request-scoped values.  
**904.** Write a Go program to parse CSV data and handle malformed rows.  
**905.** How do you implement exponential backoff with jitter in retries?  
**906.** Write a function to reverse words in a string while maintaining the word order.  
**907.** Explain the difference between shallow copy and deep copy in Go.  
**908.** Write a function to detect if a linked list has a cycle and find the start of the cycle.  
**909.** How do you implement a rate limiter using the leaky bucket algorithm?  
**910.** Write a Go program to generate prime numbers up to n using the Sieve of Eratosthenes.  
**911.** Explain the use of sync.Cond and provide a use case example.  
**912.** Write a function to calculate the product of all elements in a slice.  
**913.** How do you use the reflect package to inspect struct tags?  
**914.** Write a program to implement a concurrent-safe map without using sync.Map.  
**915.** Explain how to write table-driven tests in Go.  
**916.** Write a function to merge two binary trees.  
**917.** How do you implement a custom HTTP client with retry logic?  
**918.** Write a Go program to monitor system resource usage (CPU, memory).  
**919.** Explain the difference between a buffered and unbuffered channel.  
**920.** Write a function to convert a slice of strings to a map for quick lookup.  
**921.** How do you use Go’s context package to implement request cancellation?  
**922.** Write a program to implement a binary search on a rotated sorted array.  
**923.** Explain the purpose of Go’s go vet tool.  
**924.** Write a function to serialize a Go struct to XML.  
**925.** How do you implement a worker pool that dynamically adjusts the number of workers?  
**926.** Write a program to implement a simple HTTP file server with range requests.  
**927.** Explain how Go handles multiple return values.  
**928.** Write a function to calculate the sum of two large numbers represented as strings.  
**929.** How do you use Go’s net/http/httptest package for testing HTTP handlers?  
**930.** Write a function to find all unique triplets in an array that sum to zero.  
**931.** Explain the difference between Go’s interface{} and empty interface.  
**932.** Write a Go program to implement a simple REST API with CRUD operations.  
**933.** How do you handle JSON encoding and decoding of embedded structs?  
**934.** Write a function to rotate a matrix by 90 degrees clockwise.  
**935.** Explain how you can optimize Go programs for concurrency.  
**936.** Write a program to implement a priority queue using the container/heap package.  
**937.** How do you use Go’s time package to parse and format dates?  
**938.** Write a function to find the intersection node of two singly linked lists.  
**939.** Explain how to safely close channels and the dangers of closing them incorrectly.  
**940.** Write a Go program to parse command-line arguments with the cobra library.  
**941.** How do you implement caching in Go with TTL (time-to-live)?  
**942.** Write a function to check if a binary tree is symmetric.  
**943.** Explain the difference between var declarations and := short declarations.  
**944.** Write a program to implement a simple graph data structure with adjacency lists.  
**945.** How do you use Go’s sync package to protect shared data?  
**946.** Write a function to calculate the number of ways to climb stairs with 1 or 2 steps.  
**947.** Explain how Go’s garbage collector works.  
**948.** Write a Go program to implement a TCP client that sends and receives messages.  
**949.** How do you handle errors in Go idiomatically?  
**950.** Write a function to check if two binary trees are identical.

**Golang Interview Coding Questions: Set 20 (951–1000)**

**951.** Write a function to perform in-order traversal of a binary tree.  
**952.** How do you implement a thread-safe singleton pattern in Go?  
**953.** Explain how to handle JSON with optional fields in Go structs.  
**954.** Write a function to find the diameter of a binary tree.  
**955.** How do you implement a priority queue with custom comparator in Go?  
**956.** Write a Go program to perform matrix multiplication.  
**957.** Explain the difference between new and make in Go.  
**958.** Write a function to check if a string is a valid palindrome ignoring non-alphanumeric characters.  
**959.** How do you use channels to implement worker pools with cancellation?  
**960.** Write a program to implement a Trie with insert and search operations.  
**961.** Explain the concept of interfaces in Go with real-world examples.  
**962.** Write a function to reverse a linked list recursively.  
**963.** How do you implement memoization in Go?  
**964.** Write a program to parse and validate email addresses using regex.  
**965.** Explain how to handle deadlocks in Go programs.  
**966.** Write a function to serialize a graph data structure.  
**967.** How do you use Go’s defer statement to ensure resource cleanup?  
**968.** Write a Go program to implement a simple DNS client.  
**969.** Explain how Go manages goroutine stacks dynamically.  
**970.** Write a function to find the longest increasing subsequence in an array.  
**971.** How do you implement a concurrent linked list?  
**972.** Write a program to convert an infix expression to postfix using stacks.  
**973.** Explain how Go handles interface nil checks.  
**974.** Write a function to calculate the median of a stream of numbers.  
**975.** How do you implement a generic binary tree in Go using generics?  
**976.** Write a Go program to implement rate limiting with a sliding window.  
**977.** Explain the difference between Go’s panic and error.  
**978.** Write a function to remove duplicates from a slice of structs based on a key field.  
**979.** How do you implement a thread-safe counter using atomic operations?  
**980.** Write a program to perform a topological sort on a directed acyclic graph.  
**981.** Explain how Go’s sync.WaitGroup works and when to use it.  
**982.** Write a function to find the kth smallest element in a BST.  
**983.** How do you handle large file processing efficiently in Go?  
**984.** Write a program to implement a simple chat client using TCP sockets.  
**985.** Explain how Go’s garbage collector handles object lifetimes.  
**986.** Write a function to generate all subsets of a set (power set).  
**987.** How do you implement a pipeline pattern with fan-in and fan-out in Go?  
**988.** Write a Go program to parse command-line flags and environment variables.  
**989.** Explain the use of Go’s unsafe.Pointer and when it should be avoided.  
**990.** Write a function to compute the next lexicographical permutation of a slice.  
**991.** How do you implement a timeout for multiple concurrent operations?  
**992.** Write a program to implement a hash map from scratch.  
**993.** Explain the difference between embedding and inheritance.  
**994.** Write a function to compute the edit distance between two strings using dynamic programming.  
**995.** How do you profile and optimize memory usage in Go applications?  
**996.** Write a Go program to implement a simple RESTful API with JWT authentication.  
**997.** Explain how Go manages package dependencies with Go modules.  
**998.** Write a function to perform level order traversal on a binary tree.  
**999.** How do you handle concurrency issues in database access with Go?  
**1000.** Write a function to implement a bloom filter with bit arrays.

**DSA WITH GOLANG**

Got it! You want a comprehensive list of 1000 Golang interview questions focused on Data Structures and Algorithms (DSA), presented in sets of 50 questions at a time, similar to the style above.

Let’s start with the **first 50 Golang DSA interview questions**, and then I can continue to generate more sets on your request. Here’s Set 1 (1–50):

**Golang DSA Interview Coding Questions: Set 1 (1–50)**

1. Implement a singly linked list in Go with insert, delete, and search operations.
2. Write a function to reverse a linked list.
3. Implement a doubly linked list and its basic operations.
4. Write a function to detect a cycle in a linked list.
5. Find the middle node of a singly linked list.
6. Implement a stack using slices in Go.
7. Implement a queue using slices in Go.
8. Write a function to check if a string has balanced parentheses.
9. Implement a queue using two stacks.
10. Write a function to sort an array using quicksort.
11. Implement mergesort for an integer slice in Go.
12. Write a function to perform binary search on a sorted slice.
13. Implement a hash map from scratch using chaining for collision handling.
14. Implement a binary tree with insert and traversal methods (preorder, inorder, postorder).
15. Write a function to check if a binary tree is a binary search tree.
16. Find the height of a binary tree.
17. Write a function to print the level order traversal of a binary tree.
18. Implement a function to find the lowest common ancestor (LCA) in a binary tree.
19. Write a function to check if two binary trees are identical.
20. Implement depth-first search (DFS) on a graph.
21. Implement breadth-first search (BFS) on a graph.
22. Detect a cycle in a directed graph using DFS.
23. Detect a cycle in an undirected graph.
24. Implement Dijkstra’s shortest path algorithm.
25. Write a function to find the number of islands in a 2D grid.
26. Implement a min heap and its operations (insert, extract-min).
27. Implement a max heap in Go.
28. Write a function to find the kth largest element in an array using a heap.
29. Implement a Trie (prefix tree) with insert and search.
30. Write a function to find all permutations of a string.
31. Implement a function to find the longest common subsequence (LCS) between two strings.
32. Write a function to solve the subset sum problem using recursion and dynamic programming.
33. Implement a function to find the longest palindromic substring in a string.
34. Write a function to merge two sorted linked lists.
35. Implement a function to flatten a nested list of integers.
36. Write a function to perform topological sort on a directed acyclic graph (DAG).
37. Implement the Fibonacci sequence using memoization.
38. Write a function to find the median of two sorted arrays.
39. Implement a function to convert a binary tree to its mirror.
40. Write a function to check if a binary tree is symmetric.
41. Implement the Kadane’s algorithm to find the maximum subarray sum.
42. Write a function to find the missing number in a sequence of n numbers.
43. Implement a function to find the first non-repeating character in a string.
44. Write a function to implement the sliding window maximum.
45. Implement a function to find the intersection of two arrays.
46. Write a function to rotate an array by k elements.
47. Implement the union-find (disjoint set) data structure.
48. Write a function to solve the “trapping rain water” problem.
49. Implement a function to find all anagrams of a string in another string.
50. Write a function to find the number of connected components in an undirected graph.
51. Implement a function to check if a string is a valid palindrome considering only alphanumeric characters.
52. Write a function to find the longest substring without repeating characters.
53. Implement a function to compute the factorial of a number using recursion.
54. Write a program to detect if two strings are anagrams of each other.
55. Implement a balanced parentheses checker using a stack.
56. Write a function to merge k sorted arrays.
57. Implement a function to convert a sorted array into a balanced binary search tree.
58. Write a function to find the depth of a node in a binary tree.
59. Implement a function to find the path with the maximum sum in a binary tree.
60. Write a function to find all subsets of a set (power set).
61. Implement a function to find the length of the longest increasing subsequence.
62. Write a function to rotate a matrix by 90 degrees clockwise.
63. Implement a function to check if two strings are one edit distance apart.
64. Write a function to find the intersection node of two singly linked lists.
65. Implement a function to perform in-order traversal of a binary search tree without recursion.
66. Write a function to find the kth smallest element in a binary search tree.
67. Implement a function to serialize and deserialize a binary tree.
68. Write a function to find all permutations of an integer array.
69. Implement a function to find the number of trailing zeros in n!.
70. Write a function to solve the coin change problem using dynamic programming.
71. Implement a function to find the maximum product subarray.
72. Write a function to find the peak element in an array (element greater than neighbors).
73. Implement a function to find the first missing positive integer in an unsorted array.
74. Write a function to calculate the number of ways to climb stairs if you can take 1 or 2 steps at a time.
75. Implement a function to check if a linked list is a palindrome.
76. Write a function to find the shortest path in a maze (2D grid) from start to finish.
77. Implement a function to check if a binary tree is height-balanced.
78. Write a function to find the majority element in an array.
79. Implement a function to find the contiguous subarray with the largest sum.
80. Write a function to implement a circular queue using an array.
81. Implement a function to convert a decimal number to binary string.
82. Write a function to check if a number is a power of two.
83. Implement a function to find the longest common prefix among an array of strings.
84. Write a function to detect if a binary tree has a root-to-leaf path with a given sum.
85. Implement a function to find the next permutation of an integer slice.
86. Write a function to implement a stack with a getMin() function that returns the minimum element in O(1) time.
87. Implement a function to evaluate a postfix expression.
88. Write a function to find the number of distinct islands in a 2D grid.
89. Implement a function to count the number of ways to make change for a given amount.
90. Write a function to find the length of the longest substring with at most k distinct characters.
91. Implement a function to find the intersection of two linked lists.
92. Write a function to find all possible combinations of k numbers out of 1…n.
93. Implement a function to check if a binary tree is a subtree of another binary tree.
94. Write a function to find the lowest common ancestor in a binary search tree.
95. Implement a function to perform zigzag level order traversal of a binary tree.
96. Write a function to solve the N-Queens problem.
97. Implement a function to find the number of ways to decode a message encoded with numbers (like “A=1, B=2”).
98. Write a function to detect if two strings are scramble strings of each other.
99. Implement a function to find the minimum window substring containing all characters of another string.
100. Write a function to implement a LFU (Least Frequently Used) cache.
101. Implement a function to check if a number is a happy number.
102. Write a function to find all the duplicate numbers in an array.
103. Implement a function to find the kth largest element in an unsorted array.
104. Write a function to calculate the power of a number using fast exponentiation.
105. Implement a function to check if a string is a valid shuffle of two other strings.
106. Write a function to convert a sorted linked list to a balanced binary search tree.
107. Implement a function to find the maximum sum path between two leaves in a binary tree.
108. Write a function to merge intervals and return the merged list.
109. Implement a function to find the diameter of a binary tree.
110. Write a function to detect if two linked lists intersect.
111. Implement a function to solve the house robber problem using dynamic programming.
112. Write a function to find the smallest missing positive integer.
113. Implement a function to compute the number of ways to reach the top of stairs with variable steps.
114. Write a function to find the maximum product of three numbers in an array.
115. Implement a function to find the maximum length of a repeated subarray.
116. Write a function to find the number of islands in a grid (using BFS/DFS).
117. Implement a function to validate if a binary tree is a binary search tree.
118. Write a function to serialize and deserialize a graph.
119. Implement a function to find the minimum path sum in a grid.
120. Write a function to find the longest substring with no more than k distinct characters.
121. Implement a function to check if a string matches a pattern with wildcard characters.
122. Write a function to find the longest valid parentheses substring.
123. Implement a function to find the median of two sorted arrays.
124. Write a function to count the number of prime numbers less than a given number.
125. Implement a function to solve the word break problem using dynamic programming.
126. Write a function to find the minimum number of coins to make a certain amount.
127. Implement a function to detect a cycle in a linked list and return the node where the cycle begins.
128. Write a function to reverse nodes in k-group in a linked list.
129. Implement a function to rotate an array to the right by k steps.
130. Write a function to find the maximum XOR of two numbers in an array.
131. Implement a function to perform string compression.
132. Write a function to find the maximum depth of a nested list.
133. Implement a function to find the longest increasing path in a matrix.
134. Write a function to check if a given Sudoku board is valid.
135. Implement a function to solve the combination sum problem.
136. Write a function to find the top k frequent elements in an array.
137. Implement a function to find the skyline of a set of buildings.
138. Write a function to find the maximum sum of a rectangle in a 2D matrix.
139. Implement a function to calculate the number of trailing zeros in n factorial.
140. Write a function to find the largest rectangle in a histogram.
141. Implement a function to convert a Roman numeral to an integer.
142. Write a function to find the kth smallest element in a sorted matrix.
143. Implement a function to solve the edit distance problem.
144. Write a function to find all palindrome partitions of a string.
145. Implement a function to solve the word ladder problem.
146. Write a function to implement a circular linked list and its operations.
147. Implement a function to find the longest consecutive sequence in an array.
148. Write a function to find the majority element using Boyer-Moore voting algorithm.
149. Implement a function to perform in-place array partitioning.
150. Write a function to find the number of subarrays with sum equals k.
151. Implement a function to check if a string is a valid IPv4 address.
152. Write a function to find the longest substring where each character appears at least k times.
153. Implement a function to find the maximum sum rectangle in a 2D matrix.
154. Write a function to convert a binary number represented as a string to decimal.
155. Implement a function to find the minimum number of jumps to reach the end of an array.
156. Write a function to perform zigzag conversion of a string.
157. Implement a function to find the maximum profit from stock prices with cooldown.
158. Write a function to find the shortest transformation sequence from start to end word (word ladder).
159. Implement a function to check if a string matches a regular expression pattern (supporting '.' and '\*').
160. Write a function to generate all valid combinations of n pairs of parentheses.
161. Implement a function to find the longest substring with at most two distinct characters.
162. Write a function to detect if a linked list is a palindrome.
163. Implement a function to find the diameter of an n-ary tree.
164. Write a function to find the minimum window substring containing all characters of another string.
165. Implement a function to serialize and deserialize a binary search tree.
166. Write a function to find the sum of all left leaves in a binary tree.
167. Implement a function to solve the stone game problem.
168. Write a function to implement a queue using stacks.
169. Implement a function to find the longest substring of unique characters.
170. Write a function to find the maximum depth of a binary tree iteratively.
171. Implement a function to find the number of ways to decode a digit string.
172. Write a function to find the minimum number of deletions to make a string palindrome.
173. Implement a function to merge two sorted arrays in-place.
174. Write a function to find the maximum length of repeated subarray between two arrays.
175. Implement a function to check if two strings are scramble strings of each other.
176. Write a function to implement a binary search on a rotated sorted array.
177. Implement a function to find the number of distinct subsequences of a string.
178. Write a function to find the maximum area of water container.
179. Implement a function to find the maximum sum of subarrays of size k.
180. Write a function to flatten a binary tree to a linked list in-place.
181. Implement a function to find the minimum path sum from top to bottom in a triangle array.
182. Write a function to implement a LRU cache.
183. Implement a function to find the minimum cost to climb stairs.
184. Write a function to check if a binary tree is symmetric iteratively.
185. Implement a function to find the lowest common ancestor in a binary tree.
186. Write a function to count the number of connected components in an undirected graph.
187. Implement a function to find all subsets of a given set.
188. Write a function to detect cycle in a graph using union-find.
189. Implement a function to calculate the skyline of buildings.
190. Write a function to solve the gas station problem.
191. Implement a function to find the first missing positive integer in O(n) time.
192. Write a function to find the maximum sum path in a grid.
193. Implement a function to perform preorder traversal of a binary tree iteratively.
194. Write a function to find the kth smallest element in a sorted matrix.
195. Implement a function to find the topological order of a DAG.
196. Write a function to find all possible letter combinations of a phone number.
197. Implement a function to find the maximum width of a binary tree.
198. Write a function to find the number of trailing zeros in n factorial.
199. Implement a function to check if two binary trees are isomorphic.
200. Write a function to implement flood fill algorithm on a 2D grid.
201. Implement a function to find the minimum spanning tree of a graph using Kruskal’s algorithm.
202. Write a function to find the minimum spanning tree of a graph using Prim’s algorithm.
203. Implement a function to check if a number is a perfect square without using sqrt.
204. Write a function to calculate the square root of a number using binary search.
205. Implement a function to solve the “largest rectangle in histogram” problem.
206. Write a function to find the maximum product of two integers in an array.
207. Implement a function to find all paths from root to leaf in a binary tree.
208. Write a function to check if a binary tree is complete.
209. Implement a function to find the second largest element in a BST.
210. Write a function to find the majority element in an array (appears more than n/2 times).
211. Implement a function to detect if a linked list has a cycle using Floyd’s cycle-finding algorithm.
212. Write a function to find the minimum number of operations to convert one string into another (edit distance).
213. Implement a function to find the maximum sum of a contiguous subarray using Kadane’s algorithm.
214. Write a function to merge two binary search trees into a balanced BST.
215. Implement a function to find the lowest common ancestor in a binary search tree.
216. Write a function to perform in-order traversal of a binary tree using a stack.
217. Implement a function to convert a binary tree into a doubly linked list.
218. Write a function to find the kth largest element in a stream.
219. Implement a function to find the number of nodes at the kth level of a binary tree.
220. Write a function to implement a stack that supports push, pop, and retrieving the minimum element in constant time.
221. Implement a function to find the longest palindromic substring in a given string.
222. Write a function to check if a string can be segmented into space-separated dictionary words.
223. Implement a function to find the maximum width ramp in an array.
224. Write a function to implement preorder traversal of a binary tree recursively.
225. Implement a function to find the longest common prefix among a list of strings.
226. Write a function to find all combinations of numbers that sum to a target.
227. Implement a function to find the number of islands in a 2D grid using DFS.
228. Write a function to find the first unique character in a string.
229. Implement a function to find the maximum depth of a binary tree recursively.
230. Write a function to implement a bloom filter with basic insert and lookup.
231. Implement a function to detect and remove a loop in a linked list.
232. Write a function to find the length of the longest consecutive sequence in an unsorted array.
233. Implement a function to generate all subsets of a string including duplicates.
234. Write a function to implement a queue with two stacks.
235. Implement a function to find the sum of digits of a number until it becomes a single digit.
236. Write a function to find the first bad version in a series of versions.
237. Implement a function to find the maximum sum path from root to leaf in a binary tree.
238. Write a function to solve the knapsack problem using dynamic programming.
239. Implement a function to find the shortest path in a weighted graph using Bellman-Ford algorithm.
240. Write a function to find the kth smallest element in a BST iteratively.
241. Implement a function to calculate the number of distinct ways to climb stairs with steps of 1 or 2.
242. Write a function to reverse words in a string.
243. Implement a function to convert an integer to its English words representation.
244. Write a function to find the minimum number of moves to equal array elements.
245. Implement a function to find the maximum area of an island in a 2D grid.
246. Write a function to find the number of paths in a matrix from top-left to bottom-right with obstacles.
247. Implement a function to check if a linked list is a palindrome using O(1) space.
248. Write a function to implement a min stack supporting push, pop, and getMin in O(1).
249. Implement a function to find the median in a data stream.
250. Write a function to find all unique triplets in an array that sum to zero.
251. Implement a function to find the longest palindromic subsequence in a string.
252. Write a function to implement a priority queue using a heap.
253. Implement a function to find the number of connected components in an undirected graph using union-find.
254. Write a function to find the first missing positive integer in an unsorted array in O(n) time and O(1) space.
255. Implement a function to find the maximum sum submatrix in a 2D matrix.
256. Write a function to check if a given string is an interleaving of two other strings.
257. Implement a function to solve the Sudoku puzzle using backtracking.
258. Write a function to find the minimum edit distance between two strings using dynamic programming.
259. Implement a function to perform a level order traversal of a binary tree.
260. Write a function to implement a stack using a linked list.
261. Implement a function to detect a cycle in a directed graph using DFS and recursion stack.
262. Write a function to find the longest substring with at most k distinct characters.
263. Implement a function to solve the rod cutting problem using dynamic programming.
264. Write a function to find the maximum sum of a subarray with at most k elements removed.
265. Implement a function to convert a number to its Roman numeral representation.
266. Write a function to find the number of distinct subsequences of a string that equals another string.
267. Implement a function to check if a linked list is a palindrome using a stack.
268. Write a function to solve the combination sum II problem (each number can be used once).
269. Implement a function to find the length of the longest substring without repeating characters.
270. Write a function to find the minimum number of platforms needed for a railway station.
271. Implement a function to find the largest rectangle containing only 1's in a binary matrix.
272. Write a function to find the maximum sum path in a triangle using bottom-up dynamic programming.
273. Implement a function to convert a BST to a sorted doubly linked list.
274. Write a function to find the kth largest element in a min-heap.
275. Implement a function to find the number of ways to decode a string containing digits.
276. Write a function to find all anagrams of a pattern in a string.
277. Implement a function to solve the “largest rectangle in histogram” using a stack.
278. Write a function to check if a graph is bipartite using BFS.
279. Implement a function to find the shortest path between two nodes in an unweighted graph.
280. Write a function to implement a LRU cache using a hashmap and doubly linked list.
281. Implement a function to find all subsets that sum up to a target value.
282. Write a function to find the minimum path sum in a grid using recursion with memoization.
283. Implement a function to find the maximum number of points that lie on a straight line.
284. Write a function to find the maximum product of subarray elements.
285. Implement a function to check if a binary tree is a valid BST using inorder traversal.
286. Write a function to implement the flood fill algorithm using BFS.
287. Implement a function to count the number of islands in a grid using Union-Find.
288. Write a function to find the longest palindromic substring using expand around center.
289. Implement a function to solve the “word break II” problem to list all possible segmentations.
290. Write a function to perform inorder traversal of a binary tree using Morris traversal.
291. Implement a function to find the median of two sorted arrays of different lengths.
292. Write a function to convert a decimal number to hexadecimal representation.
293. Implement a function to find the largest sum of non-adjacent numbers in an array.
294. Write a function to find the maximum area under the histogram.
295. Implement a function to check if a number is an Armstrong number.
296. Write a function to generate Pascal’s triangle up to n rows.
297. Implement a function to find the longest increasing subsequence using dynamic programming.
298. Write a function to check if a string contains all unique characters.
299. Implement a function to perform preorder traversal of an N-ary tree.
300. Write a function to find the kth smallest element in a sorted array.
301. Implement a function to find the number of ways to reach a target sum using given numbers (with repetition allowed).
302. Write a function to implement binary search on a sorted array.
303. Implement a function to check if two binary trees are identical.
304. Write a function to find all subsets of a set that sum up to a target value.
305. Implement a function to find the maximum depth of an N-ary tree.
306. Write a function to check if a string is a valid palindrome ignoring non-alphanumeric characters.
307. Implement a function to convert a sorted array to a height-balanced BST.
308. Write a function to find the kth smallest element in a binary search tree recursively.
309. Implement a function to perform level order traversal of an N-ary tree.
310. Write a function to find the longest substring with exactly k distinct characters.
311. Implement a function to detect a cycle in a directed graph using DFS.
312. Write a function to implement a stack using two queues.
313. Implement a function to find the next permutation of a sequence of numbers.
314. Write a function to merge two sorted linked lists.
315. Implement a function to rotate a linked list to the right by k places.
316. Write a function to find the maximum profit from stock prices with at most two transactions.
317. Implement a function to find the number of trailing zeros in a factorial of n.
318. Write a function to find the median of a stream of numbers.
319. Implement a function to solve the trapping rain water problem.
320. Write a function to find the longest consecutive sequence in an unsorted array.
321. Implement a function to check if a string is a valid shuffle of two other strings.
322. Write a function to find the maximum sum of a subarray of size k.
323. Implement a function to find the kth largest element in an unsorted array using a heap.
324. Write a function to solve the N-Queens puzzle.
325. Implement a function to find the number of ways to climb stairs with steps 1, 2, or 3.
326. Write a function to find all permutations of a string.
327. Implement a function to check if a linked list is a palindrome.
328. Write a function to implement a queue using two stacks.
329. Implement a function to find the longest common subsequence between two strings.
330. Write a function to perform a binary search on a rotated sorted array.
331. Implement a function to find the maximum area of water container.
332. Write a function to find the lowest common ancestor in a binary tree.
333. Implement a function to find the maximum path sum in a binary tree.
334. Write a function to solve the coin change problem using dynamic programming.
335. Implement a function to find the longest substring without repeating characters.
336. Write a function to detect and remove a loop in a linked list.
337. Implement a function to find the minimum number of deletions to make a string palindrome.
338. Write a function to find the maximum product subarray.
339. Implement a function to find the number of islands in a 2D grid.
340. Write a function to perform preorder traversal of a binary tree without recursion.
341. Implement a function to serialize and deserialize a binary tree.
342. Write a function to find the kth smallest element in a sorted matrix.
343. Implement a function to solve the word break problem using recursion with memoization.
344. Write a function to check if a string is a valid anagram of another string.
345. Implement a function to find the longest palindromic substring using dynamic programming.
346. Write a function to implement a min stack.
347. Implement a function to find the number of distinct ways to climb stairs.
348. Write a function to find all combinations of numbers that sum to a target.
349. Implement a function to find the minimum window substring containing all characters of another string.
350. Write a function to find the largest rectangle in a histogram.
351. Implement a function to find the longest increasing path in a matrix.
352. Write a function to perform DFS traversal of a graph.
353. Implement a function to find the number of ways to make change for a given amount.
354. Write a function to check if a linked list is circular.
355. Implement a function to find the k closest points to the origin in 2D plane.
356. Write a function to detect cycle in an undirected graph using DFS.
357. Implement a function to find the maximum sum subarray with no adjacent elements.
358. Write a function to find the maximum number of points on a line.
359. Implement a function to find the maximum path sum from any node to any node in a binary tree.
360. Write a function to find all permutations of a list of numbers.
361. Implement a function to perform BFS traversal of a graph.
362. Write a function to implement a Trie (Prefix Tree) with insert, search, and startsWith methods.
363. Implement a function to find the longest substring with at most k distinct characters using sliding window.
364. Write a function to find the minimum number of swaps required to sort an array.
365. Implement a function to find the number of connected components in a directed graph.
366. Write a function to find the number of subarrays with sum equal to k.
367. Implement a function to reverse a linked list recursively.
368. Write a function to find the topological sort of a graph.
369. Implement a function to find the diameter of a binary tree.
370. Write a function to find the longest common prefix among an array of strings.
371. Implement a function to convert a sorted linked list to a height-balanced BST.
372. Write a function to find the median of two sorted arrays of different sizes.
373. Implement a function to find the minimum window substring that contains all characters of another string.
374. Write a function to find the kth smallest element in an unsorted array using QuickSelect.
375. Implement a function to find all subsets of a set.
376. Write a function to find the maximum product of three numbers in an array.
377. Implement a function to check if a binary tree is symmetric recursively.
378. Write a function to implement a priority queue using a heap.
379. Implement a function to find the number of islands in a grid using Union-Find.
380. Write a function to find the longest palindromic substring using expand around center approach.
381. Implement a function to solve the N-Queens problem using backtracking.
382. Write a function to find the maximum profit in stock trading with k transactions.
383. Implement a function to serialize and deserialize a binary tree.
384. Write a function to find the maximum subarray sum using divide and conquer approach.
385. Implement a function to find the length of the longest valid parentheses substring.
386. Write a function to find the top k frequent elements in an array.
387. Implement a function to detect a cycle in a linked list and return the node where cycle begins.
388. Write a function to rotate an array to the left by k steps.
389. Implement a function to find the first missing positive integer in an unsorted array.
390. Write a function to check if a number is a power of two.
391. Implement a function to find the maximum length of a repeated subarray between two arrays.
392. Write a function to implement a min heap.
393. Implement a function to find the number of unique paths in a grid with obstacles.
394. Write a function to find all anagrams of a string within another string.
395. Implement a function to perform inorder traversal of a binary tree using recursion.
396. Write a function to find the longest substring without repeating characters using sliding window.
397. Implement a function to merge k sorted linked lists.
398. Write a function to find the minimum number of coins required to make a certain amount.
399. Implement a function to find the largest rectangle of 1’s in a binary matrix.
400. Write a function to find the sum of digits until the result is a single digit (digital root).
401. Implement a function to find the longest substring with all distinct characters.
402. Write a function to determine if two strings are anagrams of each other.
403. Implement a function to find the minimum number of jumps to reach the end of an array.
404. Write a function to find the maximum sum of a subarray of size k.
405. Implement a function to find the kth smallest element in a sorted matrix.
406. Write a function to reverse a linked list iteratively.
407. Implement a function to find the number of connected components in an undirected graph.
408. Write a function to check if a binary tree is height-balanced.
409. Implement a function to find all root-to-leaf paths in a binary tree.
410. Write a function to check if a string matches a given regex pattern with '.' and '\*' support.
411. Implement a function to find the lowest common ancestor in a BST.
412. Write a function to perform a breadth-first search in a graph.
413. Implement a function to solve the subset sum problem.
414. Write a function to merge two sorted arrays.
415. Implement a function to find the maximum depth of a binary tree.
416. Write a function to check if a linked list contains a cycle.
417. Implement a function to find the top k frequent words in a list.
418. Write a function to perform quicksort on an array.
419. Implement a function to find the median of a data stream.
420. Write a function to perform binary search on a rotated sorted array.
421. Implement a function to find all permutations of a string.
422. Write a function to convert a Roman numeral to an integer.
423. Implement a function to find the minimum number of edits required to convert one string to another.
424. Write a function to find the maximum product subarray.
425. Implement a function to solve the N-Queens problem.
426. Write a function to detect if a graph is bipartite.
427. Implement a function to perform a depth-first search on a graph.
428. Write a function to find the kth largest element in an array.
429. Implement a function to find the maximum area of a histogram.
430. Write a function to perform inorder traversal of a binary tree without recursion.
431. Implement a function to check if two binary trees are identical.
432. Write a function to find the number of unique paths in a grid.
433. Implement a function to solve the coin change problem.
434. Write a function to find the maximum sum of a path in a triangle.
435. Implement a function to check if a string is a palindrome.
436. Write a function to implement a LRU cache.
437. Implement a function to reverse words in a string.
438. Write a function to find the number of islands in a 2D grid.
439. Implement a function to serialize and deserialize a binary tree.
440. Write a function to find the longest palindromic substring.
441. Implement a function to find the first missing positive integer.
442. Write a function to find the maximum subarray sum using Kadane’s algorithm.
443. Implement a function to find the topological order of a directed acyclic graph.
444. Write a function to detect and remove loops in a linked list.
445. Implement a function to find all anagrams in a string.
446. Write a function to find the minimum number of platforms needed for trains.
447. Implement a function to find the number of trailing zeros in factorial.
448. Write a function to find the kth smallest element in a BST.
449. Implement a function to find the diameter of a binary tree.
450. Write a function to solve the word ladder problem.
451. Implement a function to find the maximum XOR of two numbers in an array.
452. Write a function to find the longest substring without repeating characters using sliding window.
453. Implement a function to find the maximum sum rectangle in a 2D matrix.
454. Write a function to check if a binary tree is a subtree of another binary tree.
455. Implement a function to find the smallest window in a string containing all characters of another string.
456. Write a function to find the maximum depth of an N-ary tree.
457. Implement a function to find the minimum number of deletions to make two strings anagrams.
458. Write a function to solve the rod cutting problem using dynamic programming.
459. Implement a function to find the kth largest element in a min-heap.
460. Write a function to convert a BST to a sorted doubly linked list.
461. Implement a function to find the number of ways to decode a message containing digits.
462. Write a function to find the lowest common ancestor in a binary tree.
463. Implement a function to find all permutations of a list of numbers.
464. Write a function to perform zigzag level order traversal of a binary tree.
465. Implement a function to find the longest increasing subsequence.
466. Write a function to check if a graph contains a cycle.
467. Implement a function to merge k sorted arrays.
468. Write a function to find the maximum profit from stock prices with cooldown period.
469. Implement a function to serialize and deserialize an N-ary tree.
470. Write a function to find the minimum edit distance between two strings.
471. Implement a function to solve the word break problem.
472. Write a function to find the maximum number of points on a line.
473. Implement a function to find the sum of two binary strings.
474. Write a function to find the maximum width of a binary tree.
475. Implement a function to find the number of connected components in a graph using DFS.
476. Write a function to find the maximum subarray sum with one deletion allowed.
477. Implement a function to find the median of two sorted arrays.
478. Write a function to implement a stack using queues.
479. Implement a function to find the longest palindromic substring using expand around center approach.
480. Write a function to detect cycle in a linked list and find the starting node of the cycle.
481. Implement a function to solve the coin change problem with minimum coins.
482. Write a function to find the maximum area of water container.
483. Implement a function to reverse a linked list between positions m and n.
484. Write a function to find the number of subarrays with sum divisible by k.
485. Implement a function to find the shortest path in a weighted graph using Dijkstra’s algorithm.
486. Write a function to find the length of the longest substring with at most two distinct characters.
487. Implement a function to generate all valid parentheses combinations for n pairs.
488. Write a function to find the first non-repeating character in a string.
489. Implement a function to solve the house robber problem.
490. Write a function to find the number of ways to climb stairs taking 1 or 2 steps.
491. Implement a function to find the maximum sum path from root to leaf in a binary tree.
492. Write a function to find the kth smallest element in a sorted matrix.
493. Implement a function to find the maximum product of two integers in an array.
494. Write a function to find all paths in a maze from start to end using backtracking.
495. Implement a function to find the maximum sum path in a triangle.
496. Write a function to find the maximum length of a subarray with equal number of 0s and 1s.
497. Implement a function to find the top k frequent elements in an array using a heap.
498. Write a function to implement a Trie (Prefix Tree).
499. Implement a function to check if a binary tree is symmetric.
500. Write a function to find the number of islands in a 2D grid using DFS.
501. Implement a function to find the maximum sum of a circular subarray.
502. Write a function to check if a linked list is a palindrome using O(1) space.
503. Implement a function to find the number of distinct islands in a 2D grid.
504. Write a function to find the lowest common ancestor in a binary search tree.
505. Implement a function to find the maximum length of a chain of pairs.
506. Write a function to find the maximum number of chunks to make sorted.
507. Implement a function to find the number of subarrays with product less than k.
508. Write a function to perform an inorder traversal of a binary tree iteratively.
509. Implement a function to find the longest substring with k distinct characters.
510. Write a function to merge intervals.
511. Implement a function to find the kth largest element in a stream of numbers.
512. Write a function to find the number of ways to paint a fence with k colors.
513. Implement a function to find the maximum points you can get by bursting balloons.
514. Write a function to find the number of palindromic substrings in a string.
515. Implement a function to find the minimum path sum in a grid.
516. Write a function to convert a binary tree to its mirror tree.
517. Implement a function to find the shortest path in a maze using BFS.
518. Write a function to find the minimum number of meeting rooms required.
519. Implement a function to find the top k largest elements in an array.
520. Write a function to solve the sliding window maximum problem.
521. Implement a function to find the minimum number of steps to convert a string to another.
522. Write a function to implement a circular queue.
523. Implement a function to find all permutations of a string with duplicate characters.
524. Write a function to solve the decode ways problem using dynamic programming.
525. Implement a function to find the maximum profit from stock prices with unlimited transactions.
526. Write a function to find the longest path in a DAG (Directed Acyclic Graph).
527. Implement a function to find the minimum number of insertions to form a palindrome.
528. Write a function to find the median of two sorted linked lists.
529. Implement a function to find the maximum area of a rectangle in a binary matrix.
530. Write a function to find the number of distinct ways to climb stairs.
531. Implement a function to detect if a graph is strongly connected.
532. Write a function to find the number of subarrays with equal number of 0s and 1s.
533. Implement a function to find the next greater element for each element in an array.
534. Write a function to find the kth smallest element in an unsorted array using QuickSelect.
535. Implement a function to find the longest palindrome subsequence.
536. Write a function to find the longest valid parentheses substring.
537. Implement a function to convert a number from decimal to binary.
538. Write a function to find the minimum cost to paint houses with no two adjacent houses having the same color.
539. Implement a function to find the lowest common ancestor of two nodes in a binary tree.
540. Write a function to find the maximum XOR subarray.
541. Implement a function to perform preorder traversal of a binary tree iteratively.
542. Write a function to find the first unique character in a string.
543. Implement a function to find the number of islands in a grid using BFS.
544. Write a function to find the maximum subarray sum using Kadane’s algorithm.
545. Implement a function to check if a string is a valid shuffle of two other strings.
546. Write a function to solve the climbing stairs problem with variable steps.
547. Implement a function to find the number of connected components in an undirected graph using Union-Find.
548. Write a function to reverse nodes in k-group in a linked list.
549. Implement a function to find the longest increasing subsequence in O(n log n) time.
550. Write a function to find the number of ways to partition an array into two subsets with equal sum.
551. Implement a function to find the maximum sum of two non-overlapping subarrays.
552. Write a function to find the minimum window substring containing all characters of another string.
553. Implement a function to find the maximum profit from stock trading with at most k transactions.
554. Write a function to find the number of ways to decode a numeric string.
555. Implement a function to find the minimum height trees in a graph.
556. Write a function to check if a binary tree is a subtree of another tree.
557. Implement a function to find the longest increasing path in a matrix.
558. Write a function to implement a LFU (Least Frequently Used) cache.
559. Implement a function to find the maximum rectangle of 1’s in a binary matrix.
560. Write a function to find the kth smallest element in a BST iteratively.
561. Implement a function to find the number of connected components in a graph using Union-Find.
562. Write a function to find the number of distinct subsequences of a string.
563. Implement a function to find the minimum cost to climb stairs.
564. Write a function to solve the trapping rain water problem.
565. Implement a function to find the longest common substring of two strings.
566. Write a function to find the maximum length of a substring with at most k distinct characters.
567. Implement a function to find the minimum number of coins needed to make change.
568. Write a function to perform a topological sort of a graph.
569. Implement a function to check if a linked list is a palindrome recursively.
570. Write a function to find the largest rectangle area in a histogram using a stack.
571. Implement a function to serialize and deserialize a binary tree using BFS.
572. Write a function to find the maximum profit from stock prices with cooldown.
573. Implement a function to find the diameter of a binary tree.
574. Write a function to find the kth largest element in an unsorted array using a heap.
575. Implement a function to find the number of distinct islands in a grid using DFS.
576. Write a function to find the maximum sum of a circular subarray using Kadane’s algorithm.
577. Implement a function to solve the word break problem using dynamic programming.
578. Write a function to find the lowest common ancestor of two nodes in a binary search tree.
579. Implement a function to find the longest palindromic subsequence in a string.
580. Write a function to detect cycle in a directed graph using DFS.
581. Implement a function to find the maximum sum path from root to leaf in a binary tree.
582. Write a function to find the number of ways to partition an array into two subsets with equal sum.
583. Implement a function to find the minimum number of steps to convert one string to another (Edit Distance).
584. Write a function to find all unique triplets in an array that sum to zero (3Sum problem).
585. Implement a function to check if a string contains all unique characters without using extra space.
586. Write a function to find the maximum product of three numbers in an array.
587. Implement a function to find the maximum number of points on a line.
588. Write a function to perform an iterative preorder traversal of a binary tree using a stack.
589. Implement a function to find the top k frequent elements in an array using a hash map and heap.
590. Write a function to find the minimum number of meeting rooms required given meeting time intervals.
591. Implement a function to find the maximum path sum in a binary tree.
592. Write a function to find the first missing positive integer in an unsorted array.
593. Implement a function to reverse nodes in k-group in a linked list recursively.
594. Write a function to solve the climbing stairs problem with variable steps using memoization.
595. Implement a function to find the minimum number of swaps required to sort an array.
596. Write a function to find the kth smallest element in a sorted matrix using a min-heap.
597. Implement a function to find the length of the longest valid parentheses substring.
598. Write a function to find all permutations of a string with duplicates using backtracking.
599. Implement a function to find the maximum length of a substring without repeating characters using sliding window.
600. Write a function to find the number of distinct subsequences that equal a given string.
601. Implement a function to find the maximum sum of two non-overlapping subarrays.
602. Write a function to find the minimum window substring containing all characters of another string.
603. Implement a function to find the maximum profit from stock trading with at most k transactions.
604. Write a function to find the number of ways to decode a numeric string.
605. Implement a function to find the minimum height trees in a graph.
606. Write a function to check if a binary tree is a subtree of another tree.
607. Implement a function to find the longest increasing path in a matrix.
608. Write a function to implement a LFU (Least Frequently Used) cache.
609. Implement a function to find the maximum rectangle of 1’s in a binary matrix.
610. Write a function to find the kth smallest element in a BST iteratively.
611. Implement a function to find the number of connected components in a graph using Union-Find.
612. Write a function to find the number of distinct subsequences of a string.
613. Implement a function to find the minimum cost to climb stairs.
614. Write a function to solve the trapping rain water problem.
615. Implement a function to find the longest common substring of two strings.
616. Write a function to find the maximum length of a substring with at most k distinct characters.
617. Implement a function to find the minimum number of coins needed to make change.
618. Write a function to perform a topological sort of a graph.
619. Implement a function to check if a linked list is a palindrome recursively.
620. Write a function to find the largest rectangle area in a histogram using a stack.
621. Implement a function to serialize and deserialize a binary tree using BFS.
622. Write a function to find the maximum profit from stock prices with cooldown.
623. Implement a function to find the diameter of a binary tree.
624. Write a function to find the kth largest element in an unsorted array using a heap.
625. Implement a function to find the number of distinct islands in a grid using DFS.
626. Write a function to find the maximum sum of a circular subarray using Kadane’s algorithm.
627. Implement a function to solve the word break problem using dynamic programming.
628. Write a function to find the lowest common ancestor of two nodes in a binary search tree.
629. Implement a function to find the longest palindromic subsequence in a string.
630. Write a function to detect cycle in a directed graph using DFS.
631. Implement a function to find the maximum sum path from root to leaf in a binary tree.
632. Write a function to find the number of ways to partition an array into two subsets with equal sum.
633. Implement a function to find the minimum number of steps to convert one string to another (Edit Distance).
634. Write a function to find all unique triplets in an array that sum to zero (3Sum problem).
635. Implement a function to check if a string contains all unique characters without using extra space.
636. Write a function to find the maximum product of three numbers in an array.
637. Implement a function to find the maximum number of points on a line.
638. Write a function to perform an iterative preorder traversal of a binary tree using a stack.
639. Implement a function to find the top k frequent elements in an array using a hash map and heap.
640. Write a function to find the minimum number of meeting rooms required given meeting time intervals.
641. Implement a function to find the maximum path sum in a binary tree.
642. Write a function to find the first missing positive integer in an unsorted array.
643. Implement a function to reverse nodes in k-group in a linked list recursively.
644. Write a function to solve the climbing stairs problem with variable steps using memoization.
645. Implement a function to find the minimum number of swaps required to sort an array.
646. Write a function to find the kth smallest element in a sorted matrix using a min-heap.
647. Implement a function to find the length of the longest valid parentheses substring.
648. Write a function to find all permutations of a string with duplicates using backtracking.
649. Implement a function to find the maximum length of a substring without repeating characters using sliding window.
650. Write a function to find the number of distinct subsequences that equal a given string.
651. Implement a function to find the minimum number of coins needed to make a given amount using dynamic programming.
652. Write a function to find the length of the longest increasing subsequence in an array.
653. Implement a function to detect cycle in an undirected graph using Union-Find.
654. Write a function to find the maximum sum subarray with no adjacent elements.
655. Implement a function to find all subsets of a set (power set).
656. Write a function to implement a circular buffer.
657. Implement a function to check if two strings are anagrams using a hash map.
658. Write a function to find the number of ways to climb stairs when you can take 1, 2, or 3 steps at a time.
659. Implement a function to find the longest substring with exactly k distinct characters.
660. Write a function to find the number of islands in a grid using BFS.
661. Implement a function to merge two binary search trees into a sorted list.
662. Write a function to find the maximum width of a binary tree.
663. Implement a function to find the kth smallest element in a BST recursively.
664. Write a function to find the minimum path sum from top to bottom in a triangle.
665. Implement a function to check if a binary tree is a valid BST.
666. Write a function to find the longest palindromic substring using dynamic programming.
667. Implement a function to serialize and deserialize a binary tree using preorder traversal.
668. Write a function to implement an LRU cache using a doubly linked list and hashmap.
669. Implement a function to find the maximum profit from stock trading with a transaction fee.
670. Write a function to find the median of a data stream using two heaps.
671. Implement a function to find the maximum area of water container.
672. Write a function to find the number of distinct subsequences of a string that equal another string.
673. Implement a function to perform level order traversal of a binary tree.
674. Write a function to find the minimum number of steps to convert one string to another using edit distance.
675. Implement a function to check if a linked list has a cycle using Floyd’s cycle detection algorithm.
676. Write a function to find the longest substring without repeating characters using sliding window.
677. Implement a function to find the number of connected components in an undirected graph.
678. Write a function to find the top k frequent elements in an array using bucket sort.
679. Implement a function to find the first missing positive integer in an unsorted array.
680. Write a function to find the length of the longest consecutive elements sequence in an array.
681. Implement a function to solve the house robber problem using dynamic programming.
682. Write a function to find the maximum sum path from root to leaf in a binary tree.
683. Implement a function to find the lowest common ancestor of two nodes in a binary tree.
684. Write a function to find the number of unique paths in a grid with obstacles.
685. Implement a function to find the maximum sum subarray using Kadane’s algorithm.
686. Write a function to find all anagrams of a string within another string.
687. Implement a function to reverse a linked list iteratively.
688. Write a function to implement quicksort on an array.
689. Implement a function to find the kth largest element in an unsorted array using QuickSelect.
690. Write a function to find the number of trailing zeros in n factorial.
691. Implement a function to find the maximum product subarray.
692. Write a function to solve the coin change problem using dynamic programming.
693. Implement a function to find the maximum area of a rectangle in a histogram.
694. Write a function to check if two binary trees are identical.
695. Implement a function to find the number of ways to decode a numeric string.
696. Write a function to find the longest palindrome subsequence in a string.
697. Implement a function to find the maximum width ramp in an array.
698. Write a function to implement a Trie with insert and search functionality.
699. Implement a function to find the minimum number of meeting rooms required given time intervals.
700. Write a function to solve the word ladder problem using BFS.
701. Implement a function to find the maximum sum of a subarray with length exactly k.
702. Write a function to find the minimum number of steps to reach the last index in an array (jump game).
703. Implement a function to find the number of ways to climb stairs with variable steps.
704. Write a function to find the lowest common ancestor in a binary search tree iteratively.
705. Implement a function to find the maximum profit from stock prices with at most two transactions.
706. Write a function to find the longest substring where the frequency of each character is at least k.
707. Implement a function to find the maximum sum rectangle in a 2D matrix using Kadane’s algorithm.
708. Write a function to perform vertical order traversal of a binary tree.
709. Implement a function to find the maximum length of a substring without repeating characters using sliding window.
710. Write a function to find the maximum area of a histogram using stack.
711. Implement a function to find the kth smallest element in a min-heap.
712. Write a function to detect cycle in a directed graph using Kahn’s algorithm.
713. Implement a function to solve the house robber problem II (houses in a circle).
714. Write a function to find the maximum depth of an N-ary tree.
715. Implement a function to serialize and deserialize an N-ary tree.
716. Write a function to find the maximum profit from buying and selling stock with cooldown period.
717. Implement a function to check if two strings are isomorphic.
718. Write a function to find the number of islands in a 2D grid using Union-Find.
719. Implement a function to perform a zigzag level order traversal of a binary tree.
720. Write a function to find the longest palindromic substring using expand around center approach.
721. Implement a function to find the shortest path in a weighted graph using Bellman-Ford algorithm.
722. Write a function to find the number of connected components in an undirected graph using DFS.
723. Implement a function to find the minimum number of intervals to remove to make the rest non-overlapping.
724. Write a function to find the maximum product of three numbers in an array.
725. Implement a function to perform an inorder traversal of a binary tree without recursion.
726. Write a function to solve the coin change problem with infinite coins.
727. Implement a function to find the kth largest element in a BST using Morris traversal.
728. Write a function to find the first unique character in a string.
729. Implement a function to reverse nodes in k-group in a linked list iteratively.
730. Write a function to find the maximum number of points on a line.
731. Implement a function to find the longest increasing subsequence using patience sorting.
732. Write a function to find the number of ways to paint a fence with k colors and no more than two adjacent fences have the same color.
733. Implement a function to solve the jump game II (minimum number of jumps).
734. Write a function to find all permutations of a string using backtracking.
735. Implement a function to find the maximum sum path in a binary tree.
736. Write a function to find the length of the longest consecutive sequence in an unsorted array.
737. Implement a function to solve the wildcard matching problem with '?' and '\*'.
738. Write a function to find the maximum rectangle of 1’s in a binary matrix using stack.
739. Implement a function to find the minimum edit distance between two strings using dynamic programming.
740. Write a function to find the maximum sum submatrix no larger than k.
741. Implement a function to find the number of distinct islands considering rotation and reflection.
742. Write a function to find the maximum width ramp in an array.
743. Implement a function to solve the sliding window maximum problem using a deque.
744. Write a function to implement an LFU cache.
745. Implement a function to find the length of the longest valid parentheses substring using stack.
746. Write a function to find the kth smallest element in a sorted matrix using binary search.
747. Implement a function to check if a binary tree is height balanced.
748. Write a function to solve the subset sum problem using backtracking.
749. Implement a function to find the longest palindromic subsequence using memoization.
750. Write a function to find the minimum cost to paint houses with different colors and no two adjacent houses having the same color.
751. Implement a function to find the minimum window substring containing all characters of another string.
752. Write a function to find the number of unique paths in a grid moving only right and down.
753. Implement a function to find the kth largest element in a min-heap efficiently.
754. Write a function to solve the combination sum problem using backtracking.
755. Implement a function to find the maximum subarray sum with at most one deletion.
756. Write a function to detect if a graph contains a cycle using DFS.
757. Implement a function to find the maximum XOR of two numbers in an array.
758. Write a function to find the length of the longest substring with at most two distinct characters.
759. Implement a function to solve the house robber III problem on a binary tree.
760. Write a function to implement a queue using two stacks.
761. Implement a function to find the diameter of a binary tree.
762. Write a function to perform level order traversal of a binary tree using a queue.
763. Implement a function to find the median of two sorted arrays of different sizes.
764. Write a function to find the first missing positive integer in an unsorted array.
765. Implement a function to solve the trap rain water problem using two pointers.
766. Write a function to check if a binary tree is symmetric around its center.
767. Implement a function to find all anagrams of a string in another string.
768. Write a function to find the number of connected components in an undirected graph using BFS.
769. Implement a function to find the longest increasing path in a matrix using DFS with memoization.
770. Write a function to find the maximum area of water container.
771. Implement a function to merge two sorted linked lists.
772. Write a function to find the kth smallest element in a BST using recursion.
773. Implement a function to serialize and deserialize a binary tree using preorder traversal.
774. Write a function to find the longest palindromic substring using expand around center.
775. Implement a function to find the number of ways to decode a numeric string.
776. Write a function to find the maximum profit from stock prices with cooldown period.
777. Implement a function to solve the coin change problem using dynamic programming.
778. Write a function to implement a Trie (prefix tree) with insert and search methods.
779. Implement a function to reverse a linked list iteratively.
780. Write a function to find the longest valid parentheses substring.
781. Implement a function to find the maximum product subarray.
782. Write a function to find the lowest common ancestor in a binary search tree iteratively.
783. Implement a function to find the number of islands in a 2D grid using DFS.
784. Write a function to find the number of ways to partition an array into two subsets with equal sum.
785. Implement a function to check if a linked list contains a cycle and find the starting node of the cycle.
786. Write a function to find the longest substring without repeating characters using sliding window.
787. Implement a function to find the maximum width of a binary tree.
788. Write a function to perform preorder traversal of a binary tree iteratively.
789. Implement a function to find the maximum sum path from root to leaf in a binary tree.
790. Write a function to find the top k frequent elements in an array using a heap.
791. Implement a function to solve the word break problem using dynamic programming.
792. Write a function to find the maximum sum rectangle in a 2D matrix.
793. Implement a function to find the maximum subarray sum using Kadane’s algorithm.
794. Write a function to implement quicksort on an array.
795. Implement a function to find the kth smallest element in an unsorted array using QuickSelect.
796. Write a function to check if two binary trees are identical.
797. Implement a function to find the minimum number of insertions to make a string palindrome.
798. Write a function to find the number of trailing zeros in n factorial.
799. Implement a function to find the number of distinct subsequences of a string.
800. Write a function to solve the climbing stairs problem with variable steps.
801. Implement a function to find the minimum number of jumps to reach the end of an array.
802. Write a function to find the length of the longest substring with at most k distinct characters.
803. Implement a function to perform a binary search on a rotated sorted array.
804. Write a function to find the maximum product of three numbers in an array.
805. Implement a function to solve the “Word Ladder” problem using BFS.
806. Write a function to find the kth largest element in a max-heap.
807. Implement a function to find all subsets of a set using backtracking.
808. Write a function to find the longest palindromic substring using dynamic programming.
809. Implement a function to find the minimum edit distance between two strings.
810. Write a function to perform inorder traversal of a binary tree recursively.
811. Implement a function to check if a graph is bipartite using BFS.
812. Write a function to find the maximum number of points on a line.
813. Implement a function to detect a cycle in a directed graph using DFS.
814. Write a function to find the longest increasing subsequence in O(n log n) time.
815. Implement a function to find the minimum window substring containing all characters of another string.
816. Write a function to solve the “House Robber II” problem.
817. Implement a function to find the maximum rectangle of 1’s in a binary matrix.
818. Write a function to find the kth smallest element in a sorted matrix.
819. Implement a function to find the diameter of a binary tree.
820. Write a function to serialize and deserialize a binary tree using BFS.
821. Implement a function to solve the “Trapping Rain Water” problem using stacks.
822. Write a function to find the number of islands in a 2D grid using DFS.
823. Implement a function to find the maximum sum subarray using Kadane’s algorithm.
824. Write a function to implement a Trie with insert, search, and startsWith methods.
825. Implement a function to find the number of connected components in an undirected graph using Union-Find.
826. Write a function to find the kth largest element in an unsorted array using QuickSelect.
827. Implement a function to find the maximum sum path from root to leaf in a binary tree.
828. Write a function to reverse nodes in k-group in a linked list iteratively.
829. Implement a function to solve the “Word Break” problem using dynamic programming.
830. Write a function to find the longest valid parentheses substring.
831. Implement a function to find the first unique character in a string.
832. Write a function to find the minimum cost to paint houses with k colors with no two adjacent houses having the same color.
833. Implement a function to solve the “Decode Ways” problem.
834. Write a function to find the number of distinct subsequences of a string.
835. Implement a function to perform preorder traversal of a binary tree iteratively.
836. Write a function to find the minimum number of meeting rooms required given intervals.
837. Implement a function to check if a binary tree is height balanced.
838. Write a function to find the maximum XOR of two numbers in an array.
839. Implement a function to find the length of the longest consecutive elements sequence in an array.
840. Write a function to find the minimum number of swaps required to sort an array.
841. Implement a function to solve the “Sliding Window Maximum” problem using a deque.
842. Write a function to find the number of ways to partition an array into two subsets with equal sum.
843. Implement a function to find the kth smallest element in a BST using Morris traversal.
844. Write a function to detect cycle in a linked list and return the node where the cycle begins.
845. Implement a function to find the length of the longest substring without repeating characters.
846. Write a function to find the number of unique binary search trees that can be formed with n nodes.
847. Implement a function to perform level order traversal of a binary tree using recursion.
848. Write a function to find the minimum cost path in a grid from top-left to bottom-right.
849. Implement a function to find the number of ways to decode a numeric string using memoization.
850. Write a function to find the maximum sum of a circular subarray using Kadane’s algorithm.
851. Implement a function to find the longest substring with equal number of 0s and 1s.
852. Write a function to implement a min stack supporting push, pop, top, and retrieving the minimum element in O(1).
853. Implement a function to find the maximum length of a substring containing at most k distinct characters.
854. Write a function to find the kth largest element in a stream of numbers.
855. Implement a function to find the longest arithmetic subsequence in an array.
856. Write a function to find the number of subarrays with sum equal to k.
857. Implement a function to find the majority element in an array.
858. Write a function to find all combinations of well-formed parentheses given n pairs.
859. Implement a function to find the maximum gap between successive elements in sorted form.
860. Write a function to find the minimum number of refueling stops to reach a destination.
861. Implement a function to find the lowest common ancestor of a binary tree using recursion.
862. Write a function to find the number of unique paths from top-left to bottom-right in a grid with obstacles.
863. Implement a function to find the maximum area of an island in a 2D grid.
864. Write a function to find the longest substring where the frequency of each character is at least k.
865. Implement a function to find the maximum sum of two non-overlapping subarrays.
866. Write a function to implement an iterator for a binary search tree with next() and hasNext() methods.
867. Implement a function to find the number of ways to paint a fence with k colors such that no more than two adjacent fences have the same color.
868. Write a function to solve the “Palindrome Partitioning” problem using backtracking.
869. Implement a function to find the length of the longest mountain in an array.
870. Write a function to detect if two strings are one edit distance apart.
871. Implement a function to solve the “Word Search” problem on a 2D grid using DFS.
872. Write a function to find the longest palindromic subsequence in a string using bottom-up DP.
873. Implement a function to find the minimum path sum in a grid from top-left to bottom-right.
874. Write a function to find the maximum number of events that can be attended given their start and end times.
875. Implement a function to find the maximum XOR of a subarray.
876. Write a function to find the length of the longest substring with all distinct characters.
877. Implement a function to find the number of connected components in a graph using BFS.
878. Write a function to implement a binary search tree iterator.
879. Implement a function to find the kth smallest element in a max-heap.
880. Write a function to find the minimum number of steps to reduce a number to 1 using allowed operations.
881. Implement a function to check if a string matches a pattern with wildcard characters.
882. Write a function to find all permutations of a string with duplicates using backtracking.
883. Implement a function to find the maximum size square of 1’s in a binary matrix.
884. Write a function to find the first missing positive integer in an unsorted array in O(n) time and O(1) space.
885. Implement a function to find the maximum product of two elements in an array.
886. Write a function to find the minimum cost to climb stairs when you can take 1 or 2 steps at a time.
887. Implement a function to find the number of ways to reach the top of a staircase with n steps.
888. Write a function to find the kth smallest number in a multiplication table.
889. Implement a function to find the longest common prefix of an array of strings.
890. Write a function to implement a basic calculator to evaluate a simple expression string.
891. Implement a function to find the number of islands in a 2D grid using Union-Find.
892. Write a function to find the length of the longest substring without repeating characters using sliding window.
893. Implement a function to solve the “Jump Game” problem.
894. Write a function to find the maximum sum of a subarray after deleting at most one element.
895. Implement a function to find the kth largest element in a stream using a min-heap.
896. Write a function to find the maximum length of a chain of pairs.
897. Implement a function to perform a zigzag level order traversal of a binary tree.
898. Write a function to find the number of distinct subsequences of a string that equal another string.
899. Implement a function to check if a string is a valid palindrome after removing at most one character.
900. Write a function to solve the “Longest Increasing Path in a Matrix” problem.
901. Implement a function to find the minimum number of deletions to make two strings anagrams.
902. Write a function to find the maximum length of a substring with all unique characters using sliding window.
903. Implement a function to find the longest repeated substring in a string using suffix arrays.
904. Write a function to find the number of ways to climb stairs with steps of size 1, 2, or 3.
905. Implement a function to find the median of a data stream using two heaps.
906. Write a function to solve the “Decode String” problem with nested encoded patterns.
907. Implement a function to check if a linked list is a palindrome.
908. Write a function to find the maximum depth of a binary tree using DFS.
909. Implement a function to find the number of ways to partition an array into three subsets with equal sum.
910. Write a function to find the number of substrings that contain at most k distinct characters.
911. Implement a function to solve the “Maximum Swap” problem.
912. Write a function to find the minimum number of steps to convert one string to another using edit distance.
913. Implement a function to perform topological sort on a directed acyclic graph.
914. Write a function to find the longest substring where every character appears at least k times.
915. Implement a function to find the maximum profit from stock prices with cooldown period.
916. Write a function to solve the “Coin Change 2” problem (number of ways to make change).
917. Implement a function to find the maximum number of points on a line in 2D.
918. Write a function to find the length of the longest substring without repeating characters using a hash map.
919. Implement a function to find the kth smallest element in a BST using iterative inorder traversal.
920. Write a function to find the longest palindromic substring using Manacher’s algorithm.
921. Implement a function to find the lowest common ancestor in a binary tree.
922. Write a function to solve the “Word Break II” problem (all possible sentences).
923. Implement a function to find the number of distinct islands considering rotation and reflection.
924. Write a function to find the maximum length of a chain of pairs.
925. Implement a function to solve the “Longest Valid Parentheses” problem using stack.
926. Write a function to perform level order traversal of a binary tree using DFS.
927. Implement a function to find the maximum sum rectangle in a 2D matrix using Kadane’s algorithm.
928. Write a function to solve the “Sliding Window Maximum” problem using deque.
929. Implement a function to find the minimum number of meeting rooms required given intervals.
930. Write a function to find the number of connected components in an undirected graph using Union-Find.
931. Implement a function to find the maximum sum subarray with at most one deletion.
932. Write a function to find the length of the longest increasing subsequence using dynamic programming.
933. Implement a function to find the number of distinct subsequences of a string.
934. Write a function to find the minimum number of operations to convert one string to another (edit distance).
935. Implement a function to detect cycle in a directed graph using DFS.
936. Write a function to find the maximum product subarray.
937. Implement a function to find the minimum window substring containing all characters of another string.
938. Write a function to find the maximum sum path from root to leaf in a binary tree.
939. Implement a function to serialize and deserialize a binary tree using preorder traversal.
940. Write a function to find the longest increasing path in a matrix using DFS with memoization.
941. Implement a function to find the kth smallest element in a sorted matrix using binary search.
942. Write a function to find the maximum number of events that can be attended given their intervals.
943. Implement a function to solve the “Jump Game II” problem (minimum number of jumps).
944. Write a function to find the number of subarrays with sum divisible by k.
945. Implement a function to find the longest palindromic subsequence using memoization.
946. Write a function to find the minimum cost to paint houses with different colors with no two adjacent houses having the same color.
947. Implement a function to find the number of ways to decode a numeric string using memoization.
948. Write a function to solve the “House Robber III” problem on a binary tree.
949. Implement a function to find the number of connected components in a graph using BFS.
950. Write a function to find the kth largest element in a stream of numbers using a min-heap.
951. Implement a function to find the length of the longest substring with exactly k distinct characters.
952. Write a function to find the maximum sum of a submatrix no larger than k.
953. Implement a function to find the minimum number of platforms required for a railway station.
954. Write a function to perform preorder traversal of a binary tree using Morris traversal.
955. Implement a function to find the number of ways to partition a string into palindromic substrings.
956. Write a function to solve the “Split Array Largest Sum” problem.
957. Implement a function to find the maximum path sum in a binary tree.
958. Write a function to detect if a linked list is a palindrome using constant space.
959. Implement a function to find the kth smallest element in an unsorted array using a heap.
960. Write a function to find the maximum length of a substring containing at most k distinct characters using sliding window.
961. Implement a function to find the longest common subsequence of two strings using DP.
962. Write a function to solve the “Egg Drop” problem with minimum number of trials.
963. Implement a function to find the minimum cost to climb stairs with variable costs.
964. Write a function to find the number of connected components in a graph using DFS.
965. Implement a function to find the longest palindrome substring in a string using dynamic programming.
966. Write a function to find the number of islands in a 2D grid using BFS.
967. Implement a function to find the kth largest element in a stream using balanced BST or heaps.
968. Write a function to find the minimum edit distance between two strings using DP.
969. Implement a function to check if a binary tree is height-balanced.
970. Write a function to find the maximum product of three numbers in an array.
971. Implement a function to find the first unique character in a string using hash maps.
972. Write a function to find the minimum number of insertions to make a string palindrome.
973. Implement a function to find the number of distinct subsequences of a string equal to another string.
974. Write a function to find the longest substring without repeating characters using sliding window.
975. Implement a function to solve the “Jump Game” problem using greedy algorithm.
976. Write a function to find the maximum sum of a circular subarray.
977. Implement a function to find the number of ways to decode a numeric string using DP.
978. Write a function to implement a binary tree iterator with next and hasNext functions.
979. Implement a function to find the lowest common ancestor of two nodes in a binary tree.
980. Write a function to find the minimum number of swaps required to sort an array.
981. Implement a function to perform zigzag level order traversal of a binary tree.
982. Write a function to find the maximum length of a chain of pairs.
983. Implement a function to detect cycle in a directed graph using DFS.
984. Write a function to find the kth smallest element in a BST using recursion.
985. Implement a function to find the maximum width of a binary tree.
986. Write a function to find the number of connected components in an undirected graph using Union-Find.
987. Implement a function to find the maximum profit from stock prices with cooldown period.
988. Write a function to find the maximum sum rectangle in a 2D matrix using Kadane’s algorithm.
989. Implement a function to find the number of distinct islands in a grid.
990. Write a function to solve the “House Robber II” problem using dynamic programming.
991. Implement a function to reverse nodes in k-group in a linked list.
992. Write a function to find the maximum XOR of two numbers in an array.
993. Implement a function to find the longest increasing subsequence in O(n log n) time.
994. Write a function to find the length of the longest valid parentheses substring.
995. Implement a function to solve the “Coin Change” problem using memoization.
996. Write a function to find the maximum sum path from root to leaf in a binary tree.
997. Implement a function to serialize and deserialize a binary tree.
998. Write a function to solve the “Word Ladder II” problem (all shortest transformation sequences).
999. Implement a function to find the maximum sum of two non-overlapping subarrays.
1000. Write a function to find the minimum cost to paint houses with k colors without painting adjacent houses with the same color.
1001. Implement a function to find the length of the longest substring with at most two distinct characters.
1002. Write a function to find the number of subarrays with sum equal to k.
1003. Implement a function to find the kth largest element in an unsorted array using QuickSelect.
1004. Write a function to find the maximum product subarray.
1005. Implement a function to solve the “Longest Palindromic Subsequence” problem using DP.
1006. Write a function to find the minimum window substring containing all characters of another string.
1007. Implement a function to find the number of ways to partition an array into two subsets with equal sum.
1008. Write a function to perform preorder traversal of a binary tree iteratively.
1009. Implement a function to find the maximum number of points on a line.
1010. Write a function to find the diameter of a binary tree.
1011. Implement a function to check if a linked list has a cycle and find the cycle’s entry node.
1012. Write a function to find the number of ways to climb stairs with steps of 1 or 2.
1013. Implement a function to solve the “Word Break” problem using recursion and memoization.
1014. Write a function to find the maximum sum subarray using Kadane’s algorithm.
1015. Implement a function to serialize and deserialize a binary tree using BFS.
1016. Write a function to find the kth smallest element in a BST.
1017. Implement a function to find the maximum XOR of two numbers in an array using a Trie.
1018. Write a function to find the minimum number of meeting rooms required given intervals.
1019. Implement a function to solve the “Trapping Rain Water” problem.
1020. Write a function to find the longest consecutive sequence in an unsorted array.
1021. Implement a function to find the maximum sum path in a binary tree.
1022. Write a function to implement a Trie with insert, search, and startsWith.
1023. Implement a function to solve the “House Robber III” problem on a binary tree.
1024. Write a function to find the minimum number of swaps required to sort an array.
1025. Implement a function to find the kth largest element in a stream using a min-heap.
1026. Write a function to perform level order traversal of a binary tree using recursion.
1027. Implement a function to detect if a binary tree is height-balanced.
1028. Write a function to find the longest increasing subsequence using binary search.
1029. Implement a function to find the number of islands in a 2D grid using DFS.
1030. Write a function to find the maximum length of a chain of pairs.
1031. Implement a function to find the maximum sum of a circular subarray using Kadane’s algorithm.
1032. Write a function to find the number of distinct subsequences of a string equal to another string.
1033. Implement a function to find the minimum cost to paint houses with k colors.
1034. Write a function to solve the “Decode Ways” problem using DP.
1035. Implement a function to find the maximum area of an island in a grid.
1036. Write a function to find the kth smallest element in a sorted matrix.
1037. Implement a function to find the maximum sum rectangle in a 2D matrix.
1038. Write a function to solve the “Longest Valid Parentheses” problem using stack.
1039. Implement a function to find the number of connected components in a graph using Union-Find.
1040. Write a function to solve the “Jump Game II” problem.
1041. Implement a function to find the maximum number of events that can be attended given their intervals.
1042. Write a function to find the length of the longest substring without repeating characters using sliding window.
1043. Implement a function to find the minimum number of deletions to make two strings anagrams.
1044. Write a function to find the kth smallest element in a max-heap.
1045. Implement a function to find the minimum edit distance between two strings.
1046. Write a function to find the maximum profit from stock prices with cooldown period.
1047. Implement a function to find the number of ways to decode a numeric string.
1048. Write a function to find the maximum sum of two non-overlapping subarrays.
1049. Implement a function to find the maximum XOR of a subarray.
1050. Write a function to find the minimum cost path in a grid from top-left to bottom-right.