Question1: prepare 13 instructions of clean code and example each one of them.

1) Meaningful names for example, Result= num1+num2
2) Consistent Indentation for example:
```
For (int i=0; i<n; i++){
        Print(i);
        Sum+=1
```
3) Avoid Magic Numbers : Replace hardcoded constants with named constants to improve code understanding for example use math.pi instead of 3.142
4) Keep Functions/Methods Short for example
   Def handle_request(request) instead of **Def function_used_to_handle_request(request)**
5) Use Comments Sparingly: using them to explain parts or lines of code for example:
```
function calculate_Tax(amount) {
  // Calculate tax at 10%
  return amount * 0.10.
}
```
6) Don't Repeat Yourself (DRY) : Avoid duplicating code by extracting common functionality into reusable functions or modules for example:
```
# Bad example
def calculate_area_of_square(side):
  return side * side

def calculate_area_of_rectangle(length, width):
  return length * width

# Good example
def calculate_area(length, width=None):
  if width is None:
    return length * length
  else:
    return length * width
```
7) Error Handling: Handle errors gracefully with appropriate messages or logging for example:
```
Def divide(x,y):
        If(y=0):
            print("Cannot divide by zero.");
```
8) Avoid nested code for example:
```
If(condition):
        If(condition2):
                If(condition3):
```
9) Unit Testing: Write automated unit tests for your code to ensure correctness and facilitate future changes.
10) Use Version Control: Utilize version control systems like Git to track changes and collaborate effectively.

11) Optimize Loops: Make loop operations efficient and avoid unnecessary calculations for example:

```
result = (1000 * (1000 + 1)) // 2 => use this instead of

result=0
for i in range(1000):
        result=+i
```

12) Single Responsibility Principle (SRP) each function or class should do exactly one thing for example:

```
Def sum(x,y):
        Result=x+y
```

13) Keep Code Modular: Break down complex functionalities into smaller, manageable modules.

```
function processOrder(order) {
  validateOrder(order);
  calculateTotal(order);
  generateInvoice(order);
}
```

Question2: application interface of python all libraries

1) Tkinter
2) PyQT5
3) PySide 2
4) Kivy
5) WxPython

Question3: framework (flutter) based on what? (Mobile)

It is based on the Dart programming language, which is also developed by Google. Flutter allows developers to build natively compiled applications for mobile, web, and desktop from a single codebase.

Question4: what's the language which support data typed and not typed?

Data-typed Languages: Java ,C, C++, Swift, Kotlin

Non-data typed languages: Python, Ruby, JS, PHP

Question5: what's the HashMap? what's things based on it in python language.

A HashMap is a data structure that provides a way to store and retrieve key-value pairs in an efficient manner. It is also known as an associative array, dictionary, or hash table in other programming languages. In a HashMap, keys are unique, and each key is associated with a single value. The main idea behind a HashMap is to use a hash function to convert the key into an index in an array, where the corresponding value is stored. This allows for fast retrieval of values based on their keys.

Operations that could be done with HashMap: Creating of dictionary, Adding/updating key-value pairs, Accessing values by key, Removing key, Check if key exists

Question6: how sort data which contains types of different data.

Order of Data Types: Python's default sorting order is as follows:

1. bool (True < False)
2. int
3. float
4. str

Question7: what's the language which support automatic garbage collection?

The language that supports automatic garbage collection is Java. Garbage collection is a memory management feature that automatically manages the allocation and deallocation of memory in a programming language. It helps prevent memory leaks and allows developers to focus on writing code without worrying about manual memory management.

Question8: what’s the opposite for multimap in C++ and python?

For C++: A map is a container in C++ that stores key-value pairs in a sorted order based on the keys. Each key in a map is unique, and the values associated with the keys can be accessed, modified, or erased using the keys.

For Python: dictionaries (dict) which are very flexible and allow you to store multiple values for a single key using data structures like lists, sets, or tuples. This allows you to create multimap-like behavior using the standard dictionary type.

Question9: whats the advantages of c++ 2023?

Compilation time and execution time is faster than most programming languages.

Question10: What is assembly language?

Assembly language is a low-level programming language that serves as a bridge between machine instructions and high-level languages. It finds applications in system programming, embedded systems, real-time systems, and performance-critical tasks like graphics rendering and cryptography. It is used in reverse engineering, hardware interfacing, and teaching computer architecture concepts. However, due to its complexity and lack of portability, its use has diminished in favor of higher-level languages for most general-purpose application development.

Question 11: Difference between design pattern and architecture pattern?

Design Pattern:

A design pattern is a reusable solution to a specific coding problem within a particular context. It is focused on the design of individual classes and objects and addresses issues related to object creation, composition, and interaction. Design patterns help to make code more flexible, maintainable, and scalable by promoting best practices and standardizing solutions to recurring design challenges. Examples of design patterns include the Singleton pattern, Factory pattern, Observer pattern, and Adapter pattern.

Architectural Pattern:

An architectural pattern is a high-level design that provides a blueprint for organizing an entire software system. It deals with the overall structure, organization, and flow of the application, defining the relationships between major components and modules. Architectural patterns guide the overall software structure, making it easier to manage complexity, achieve modularity, and promote maintainability and scalability. Examples of architectural patterns include the Model-View-Controller (MVC) pattern, Microservices architecture, Layered architecture, and Event-Driven architecture.

Question12: how make infinity loop ?

```
While(true):

        Print(“Hello”)
```

Question13: Comparison between Kivy-React-Flutter

	Kivy	React	Flutter
Language	Python	JS	Dart
Target Platform	Cross-Platform	ReactJs(Web), ReactNative(Mobile), Desktop(Electron)	Cross-platform
UI Dev	Declarative (KV) and imperative (Python)	Declarative (JSX)	Declarative (Dart)
Rendering Engine	OpenGL	Virtual DOM	Skia

Question14: Drop and take in python.

"Drop" typically refers to the operation of skipping a specified number of elements from the beginning of a collection and returning the remaining elements. It is used in functional programming to create a new collection with certain elements removed from the original collection.

Take" typically refers to the operation of selecting a specified number of elements from the beginning of a collection and returning them as a new collection. Like "drop," it is used in functional programming to create a new collection with specific elements from the original collection.

Question15: what is Parsing the source in design pattern with example?

Parsing, syntax analysis or syntactic analysis is the process of analysing a string of symbols, either in natural language, computer languages or data structures, conforming to the rules of a formal grammar. parsing refers to the process of converting a sequence of characters (usually text) into a structured data representation. It is a common task in software development when dealing with input data, configuration files, network protocols, or any scenario where raw textual data needs to be transformed into a more organized and usable format. The Interpreter pattern is a design pattern that relates to parsing. It provides a way to interpret or evaluate sentences in a language. The pattern is commonly used to create domain-specific languages or to parse and interpret complex expressions or rules.

Question16: What is paging?

in the context of computer memory management, paging is a memory management scheme that allows the physical memory (RAM) to be divided into fixed-size blocks called "frames," and the logical memory (the memory as seen by the processes running on the system) is divided into fixed-size blocks called "pages." Paging enables the mapping of logical addresses to physical addresses, allowing processes to access memory efficiently and independently of the actual physical memory layout. The main idea behind paging is to eliminate the need for contiguous allocation of memory for processes. Instead of allocating a continuous block of memory to each process, the memory is divided into fixed-size frames, and the logical memory of each process is divided into fixed-size pages. These pages can be mapped to any available frame in physical memory. This allows for better memory utilization and efficient memory sharing among multiple processes.

Question 17: Fragmentation in OS ?

Fragmentation in the context of operating systems refers to the inefficient use of memory or storage space, resulting in unused or partially filled memory blocks or disk sectors. There are two main types of fragmentation:

1) Memory Fragmentation: Memory fragmentation occurs when the available memory in a computer system is divided into small, non-contiguous blocks, and there are unused gaps between allocated blocks.
2) Disk Fragmentation : Disk fragmentation occurs on storage devices like hard drives. When files are created, modified, or deleted over time, the data can become scattered across different physical sectors on the disk.

Question 18: How to set priority in quote in python?

In Python, you can set the priority of quotes when defining strings. Python allows you to use either single quotes (') or double quotes (") to create string literals, and both have the same functionality. However, using different quotes inside a string can be helpful to avoid escaping characters in some cases.

Question 19: Types of CPU Scheduling?

1) First-Come, First-Served (FCFS):

Processes are executed in the order they arrive in the ready queue. The first process that enters the queue is the first one to get the CPU. It suffers from the "convoy effect," where long processes can delay the execution of short processes behind them.

2) Shortest Job Next (SJN) / Shortest Job First (SJF):

The process with the shortest burst time is selected for execution next. It aims to minimize average waiting time and can lead to optimal scheduling for non-preemptive cases. However, predicting the burst time in real systems is challenging.

3) Priority Scheduling:

Each process is assigned a priority, and the CPU selects the process with the highest priority for execution. It can be either preemptive or non-preemptive. Preemptive priority scheduling can lead to starvation for lower priority processes.

4) Round Robin (RR):

Processes are executed in a circular queue, with a fixed time quantum (time slice) allocated to each process. After the time quantum expires, the process is moved to the back of the queue. It provides fair allocation of CPU time to all processes but can result in higher context switching overhead.