ASSIGNMENT No: 1

CODE:

```
# factorial.proto file
to define the gRPC service
syntax = "proto3";
service FactorialService {
   rpc Calculate (FactorialRequest) returns (FactorialResponse);}
message FactorialRequest {
   int32 n = 1;}
message FactorialResponse {
   int64 factorial = 1;}


# Server.py code
import grpc
from concurrent import futures
import factorial_pb2
import factorial_pb2_grpc
class FactorialServicer(factorial_pb2_grpc.FactorialServiceServicer):
   def Calculate(self, request, context):
      n = request.n
      if n < 0:
         context.abort(grpc.StatusCode.INVALID_ARGUMENT, "n must be non-negative")
      factorial = 1
      for i in range(2, n + 1):
         factorial *= i
      return factorial_pb2.FactorialResponse(factorial=factorial)


def serve():
   server = grpc.server(futures.ThreadPoolExecutor(max_workers=4))
   factorial_pb2_grpc.add_FactorialServiceServicer_to_server(FactorialServicer(), server)
```

```
    server.add_insecure_port("[::]:50051")
    server.start()
    print("Server started on port 50051")
    server.wait_for_termination()
if __name__ == "__main__":
    serve()


# Client.py code
import grpc
import factorial_pb2
import factorial_pb2_grpc
def get_factorial(n):
    with grpc.insecure_channel("localhost:50051") as channel:
        stub = factorial_pb2_grpc.FactorialServiceStub(channel)
        response = stub.Calculate(factorial_pb2.FactorialRequest(n=n))
        return response.factorial
if __name__ == "__main__":
    n = int(input("Enter a non-negative integer: "))
    factorial = get_factorial(n)
    print(f"Factorial of {n} is {factorial}")
```

Output :

Server.py

```
PS C:\Users\Shree\Desktop\Cl 3 pratical code\CI1> python server.py
Server started on port 50051
```

Client.py

```
PS C:\Users\Shree\Desktop\Cl 3 pratical code\CI1> python client.py
Enter a non-negative integer: 5
Factorial of 5 is 120
```

ASSIGNMENT No: 2

CODE:

Concatenation.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Concatenation extends Remote {
   String concatenateStrings(String str1, String str2) throws RemoteException;
}
```

ConcatenationServer.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;

public class ConcatenationServer extends UnicastRemoteObject implements Concatenation {

   // ✅ Correct Constructor with 'throws RemoteException'
   public ConcatenationServer() throws RemoteException {
      super();
   }

   @Override
   public String concatenateStrings(String str1, String str2) throws RemoteException {
      return str1 + str2;
   }

   public static void main(String[] args) {
      try {
         // ✅ Start RMI registry in code instead of manually
         LocateRegistry.createRegistry(1099);
         System.out.println("RMI Registry started...");

         ConcatenationServer server = new ConcatenationServer();
         Naming.rebind("rmi://localhost/ConcatenationService", server);
         System.out.println("Concatenation Server is running...");
      } catch (Exception e) {
         e.printStackTrace(); // Print full error stack trace
      }
   }
}
```

ConcatenationClient.java

```
import java.rmi.Naming;

public class ConcatenationClient {
    public static void main(String[] args) {
        try {
            Concatenation obj = (Concatenation)
Naming.lookup("rmi://localhost/ConcatenationService");

            // Example Input
            String str1 = "Hello, ";
            String str2 = "World!";

            // Remote Method Call
            String result = obj.concatenateStrings(str1, str2);

            System.out.println("Concatenated Result: " + result);
        } catch (Exception e) {
            System.err.println("Client Error: " + e.getMessage());
        }
    }
}
```

OUTPUT:

ConcatenationServer.java

```
PS C:\Users\Shree\Desktop\Cl 3 pratical code\CI2> java ConcatenationServer
RMI Registry started...
Concatenation Server is running...
```

ConcatenationClient.java

```
PS C:\Users\Shree\Desktop\Cl 3 pratical code\CI2> java ConcatenationClient
Concatenated Result: Hello, World!
```

ASSIGNMENT No: 3

CODE:

```
import sys

# Mapper for Character Count
def char_mapper():
    for line in sys.stdin:
        line = line.strip()
        for char in line:
            print(f"{char}\t1")

# Reducer for Character Count
def char_reducer():
    current_char = None
    current_count = 0

    for line in sys.stdin:
        char, count = line.strip().split('\t')
        count = int(count)

        if current_char == char:
            current_count += count
        else:
            if current_char:
                print(f"{current_char}\t{current_count}")
            current_char = char
            current_count = count

    if current_char:
        print(f"{current_char}\t{current_count}")
```

```
# Mapper for Word Count
def word_mapper():
    for line in sys.stdin:
        line = line.strip()
        words = line.split()
        for word in words:
            print(f"{word}\t1")

# Reducer for Word Count
def word_reducer():
    current_word = None
    current_count = 0

    for line in sys.stdin:
        word, count = line.strip().split('\t')
        count = int(count)

        if current_word == word:
            current_count += count
        else:
            if current_word:
                print(f"{current_word}\t{current_count}")
            current_word = word
            current_count = count

    if current_word:
        print(f"{current_word}\t{current_count}")

# Execution based on argument passed
```

```
if __name__ == "__main__":
    if sys.argv[1] == "char_mapper":
        char_mapper()
    elif sys.argv[1] == "char_reducer":
        char_reducer()
    elif sys.argv[1] == "word_mapper":
        word_mapper()
    elif sys.argv[1] == "word_reducer":
        word_reducer()
```

input.txt

This is an apple. Apple is red in color.

OUTPUT:

```
A       1
T       1
a       3
c       1
d       1
e       3
h       1
i       4
l       2
n       2
o       1
p       3
r       2
s       2
t       1
.       2
```

```
This    1
is      2
an      1
apple   1
Apple   1
red     1
in      1
color   1
```