**Write programs for the following in a structured way:**

1. Check whether a given string is palindrome or not using Time Complexity / HashMap
2. Factorial of a number using Recursion
3. Write any two Searching and Sorting algorithms of your choice and tell about the pros and cons.
4. Build REST APIs

1. Create users

2. Read user data

3. Update user data

4. Delete a few attributes of user data and

5. Delete user records completely

1. Create a table
   1. with (ID, Phy, Chem, Maths) with ID as primary key.
   2. Add another column Biology
   3. Create 4 records
   4. Change data for a record or two
2. Find Operation

1. Find the ID with highest Phy score

2. Find the ID with highest total score

1. Sorting Operations

3. Ascending order based on total score

4. Descending order based on total score

1. Implement the following

1. User Singup

2. User SingIn

3. Forgot password

4. API Authentication & Authorization

5. RBAC

1)  
  
public class palindrome{  
 Public static void main(String[] args){

String s=”Darshan”,nstr=””;  
Char ch;

for(int i=0;i<s.length();i++){

ch=s.charAt(i);  
nstr=ch+nstr;

}

if(s==nstr){

System.out.println(“String is an palindrome”)

}

else

System.out.println(“ String is not an palindrome”)

}

}

2)Factorial

Class Factorial{

Static int factorial(int n){

if(n==0||n==1){

Return 1;}

Return n\*factorial(n-1);

}

Public static void main(String[] args){

int ans= factorial(5);

System.out.println(“Factoriaol of 5 is:” +ans);

}

}

3)Searching and sorting algo  
a)BinarySearch:- Binary Search is an divide and conquer algorithm for finding target value within sorted array.It repeatedly divides the search space in half and compares the target value with the middle element.  
  
PROS:  
Efficient:Complexity O(log n), it is highly efficient for large sorted db.  
No extra space: Doesn’t require extra space or additional space.  
  
CONS:  
Requirement:The array must be sorted before performing the operation.  
Restriction:It can be applied to sorted data only and not unsorted data.

Int BS(inta[],l,r,T){

Int m;  
while(l<r)  
{

m=l+(r-l)/2;

if(a[m]==T){ return m;}

if(a[m]<T){

l=m+1}

else{r=m-1}

}

}

b) Merge Sort:It is an divide and conquer sorting based algo that divides the input array in two halves , recursively sorts them and merges the sorted halves. It divides the unsorted arrays into equal halves until each subarray contains only one element. Then it repeatedly merges the subarrays in sorted manner.

Pros:

Stability: It is stable sorting algo and preserves relative data.

Predictable Performace:It guarantees a time complexityb of O(log n)

Cons:\

Extra space:Extra space is required to hold the temporary sub arrays.  
Recursive nature:It may result in additional functional calls or methods.

4) Build REST APIs

1. Create users:   
@Entity  
Class User{

@Id

@Generated values(Strategy=GenerationType.Identiy)

private int ID;  
 private String name;  
 private String address;  
 private String mail;

Private int phone;

//Apply getter and setter methods.  
//apply to string and equals methods.  
}

2]]. Read user data

@Repository

public interface UserRepository extends JPARepository<User, int>{

}  
  
 @RestController

Public class UserController{

@Autowired

private UserRepository userRepository;

@GetMapping(“/{id}”)

Public User readUser(@PathVariable int id){

Return userRepository.finfById(id0.orElse(null);

}

}

3]]. Update user data

@PutMapping(“/{id}”)

Public User updateUser(@PathVariable int id, @RequestBody User updatedUser){

User user=userRepository.finfById(id0;

if(user!=null){

user.setName(updateduser.getName());

user.setAddress(updateduser.getAddress());

user.setMail(updateduser.getMail());

user.setPhone(updateduser.getPhone());

return userRepository.save(user);

}

return null;

}

4]]. Delete a few attributes of user data and

@DeleteMapping(“/{id}”)

Public ResponseEntity<User> readUser(@PathVariable int id){

return userRepository.deleteById(id0.orElse(null);

}

}

5]]. Delete user records completely

@DeleteMapping

Public User readUser(){

Return userRepository.deleteAll();

}

}

5)

a) create table Subjects(id INT PRIMARY KEY AUTO\_INCREMENT, Phy INT, Chem INT,Maths INT);  
  
b) Alter table Subjects ADD COLUMN Biology INT;  
  
c) Insert into Subjects(id,Phy,Chem,Maths,Biology) values(

(1,85,95,75,65),

(2,65,89,56,75),

(3,65,65,89,95),

(4,88,77,44,66));  
  
d) UPDATE Subjects set Phy=90, Chem=80,Maths=99,Biology=88 where id=2;

UPDATE Subjects set Chem=45,Biology=45 where id=4;  
  
  
6)

a) select id from Subjects order by Phy desc LIMIT 1;  
  
b)select id from Subjects order by (Phy+Chem+Maths+Biology) desc  
 LIMIT 1;  
  
7)

a) select \* from Subjects order by (Phy+Chem+Maths+Biology) ASC;  
  
b) select \* from Subjects order by (Phy+Chem+Maths+Biology) DESC;