**JSPM’s**

Jayawantrao Sawant College of Engineering, Hadapsar.

**JAVA PROGRAMMING AND DATA**

**STRUCTURES AND**

**ALGORITHMS-IT11L**

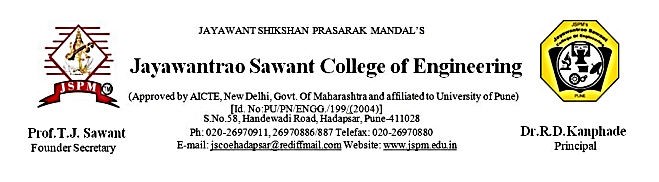
(2020 Pattern)

**Prepared By:**

**Pushkar Chandrakant Chavan**

**MCA FIRST YEAR (SEM I)**

Academic year 2023-2024



**CERTIFICATE**

This is to certify that **Mr. Pushkar Chandrakant Chavan, MCA I Year** is a bonafide student of Jayawantrao Sawant College of Engineering, Hadapsar,

Pune-28. He has successfully completed the practical work in the subject **Java Programming and Data structure and Algorithms (IT-11L)** as per the guidelines provided by Savitribai Phule Pune University for Academic Year 2023-24.

Subject Teacher Prof. Swayam Shah

1. Prof. Swayam Shah HOD
2. Prof. Krutika Kakpure

Internal Examiner External Examiner

Date: / /2023

Place: Pune

**INDEX**

# JAVA PROGRAMMING (IT11L)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.No** | **Date** | **Title** | **Page.no** | **Teacher’s**  **sign** |
| 1. | 11/09/2023 | Write a program to display transpose of matrix. |  |  |
| 2. | 11/09/2023 | Write a program to demonstrate the use of  a) Package b) Interface c) abstract class |  |  |
| 3. | 12/09/2023 | Write a program to demonstrate   1. Operations performed on String 2. Use of StringBuilder Class 3. Use of StringTokenizer Class |  |  |
| 4. | 18/09/2023 | Write a program to demonstrate user defined exception. |  |  |
| 5. | 25/09/2023 | Write a program to create a thread using a) Extending the Thread class  b) Implementing Runnable interface. |  |  |
| 6. | 26/09/2023 | Write a program to copy the contents of one  file into another file in reverse direction. |  |  |
| 7. | 3/10/2023 | Write a program to display the contents of a file. |  |  |
| 8. | 9/10/2023 | Write a program to Create Calculator by using AWT. |  |  |
| 9. | 10/10/2023 | Write a program to create a menu using AWT / Swing. |  |  |
| 10. | 16/10/2023 | Write a program to Create Log in form by using AWT/Swing and JDBC. |  |  |
| 11. | 23/10/2023 | Write a program to demonstrate operations performed on ArrayList. |  |  |
| 12. | 5/10/2023 | Write a program to demonstrate operations performed on LinkedList. |  |  |
| 13. | 16/10/2023 | Write a JDBC program to Perform CRUD  Operations on Oracle Database. (4 Different Programs) |  |  |
| 14. | 16/10/2023 | Write a program to Connect Java Application with MySQL Database. |  |  |
| 15. | 30/10/2023 | Write a servlet program to implement Get and Post methods. |  |  |
| 16. | 6/11/2023 | Write a JSP program. |  |  |

**Q1. Write a program to display transpose of matrix.**

import java.util.Scanner; public class MatrixTranspose { public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input matrix dimensions

System.out.print("Enter the number of rows: "); int rows = scanner.nextInt();

System.out.print("Enter the number of columns: "); int columns = scanner.nextInt();

// Input matrix elements int[][] matrix = new int[rows][columns]; System.out.println("Enter the matrix elements:"); for (int i = 0; i < rows; i++) { for (int j = 0; j < columns; j++) { matrix[i][j] = scanner.nextInt();

}

}

// Display original matrix

System.out.println("\nOriginal Matrix:"); displayMatrix(matrix);

// Compute and display transpose int[][] transposedMatrix = transposeMatrix(matrix); System.out.println("\nTransposed Matrix:"); displayMatrix(transposedMatrix);

}

// Function to display a matrix public static void displayMatrix(int[][] matrix) { for (int i = 0; i < matrix.length; i++) { for (int j = 0; j < matrix[0].length; j++) {

System.out.print(matrix[i][j] + " ");

}

System.out.println();

}

}

// Function to compute the transpose of a matrix public static int[][] transposeMatrix(int[][] matrix) { int rows = matrix.length;

int columns = matrix[0].length;

int[][] transposedMatrix = new int[columns][rows];

for (int i = 0; i < columns; i++) { for (int j = 0; j < rows; j++) { transposedMatrix[i][j] = matrix[j][i];

}

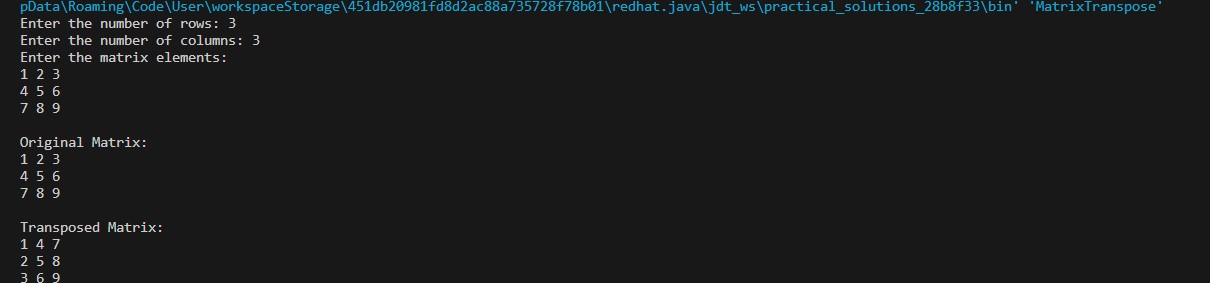
}

return transposedMatrix;

}

}

**Output:**



**Q2. Write a program to demonstrate the use of a) Package**

1. **Interface**
2. **abstract class**

A)

// File: example/MyPackageClass.java package example;

public class MyPackageClass { public void displayMessage() {

System.out.println("Hello from MyPackageClass!");

}

}

B)

// File: example/MyInterface.java package example;

public interface MyInterface { void myMethod();

}

C)

// File: example/MyAbstractClass.java package example;

public abstract class MyAbstractClass implements MyInterface { public void commonMethod() {

System.out.println("This is a common method in MyAbstractClass.");

}

// Abstract method from the interface public abstract void myMethod();

}

// File: MainProgram.java import example.\*;

public class MainProgram {

public static void main(String[] args) {

// Using the package

MyPackageClass myPackageObject = new MyPackageClass(); myPackageObject.displayMessage();

// Using the abstract class

MyAbstractClass myAbstractObject = new MyConcreteClass(); myAbstractObject.commonMethod(); myAbstractObject.myMethod();

}

}

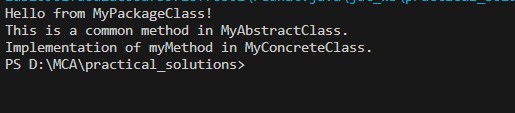
// File: example/MyConcreteClass.java package example;

public class MyConcreteClass extends MyAbstractClass { public void myMethod() {

System.out.println("Implementation of myMethod in MyConcreteClass."); }

}

**Output:**



**Q3. Write a program to demonstrate**

1. **Operations performed on String**
2. **Use of StringBuilder Class**
3. **Use of StringTokenizer Class**

a)

public class StringDemo2 { public static void main(String args[])

{

String strOb1="First String";

String strOb2="Second String";

String strOb3=strOb1;

System.out.println("Length of strOb1 : "+strOb1.length()); System.out.println("Char at index 3 in strOb1 : "+strOb1.charAt(3)); if(strOb1.equals(strOb2))

System.out.println("strOb1==strOb2"); else

System.out.println("strOb1!==strOb2"); if(strOb1.equals(strOb3))

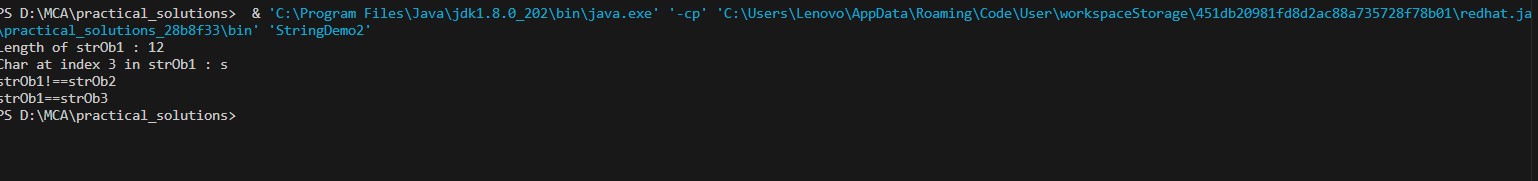
System.out.println("strOb1==strOb3"); else

System.out.println("strOb1!==strOb3");

}

}

**Output:**



b)

public class StringBuilderExample2 { public static void main(String args[])

{

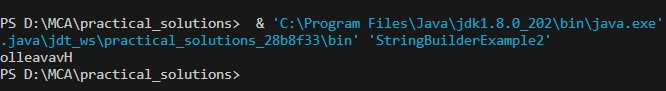
StringBuilder sb=new StringBuilder("Hello"); sb.insert(1,"Java"); sb.replace(1,3,"Java"); sb.delete(1,3); sb.reverse();

System.out.println(sb);

}

}

**Output:**

 c)

import java.util.\*; public class stringtok {

public static void main(String args[])

{

StringTokenizer st=new StringTokenizer("my name is pushkar"," "); while(st.hasMoreTokens())

{

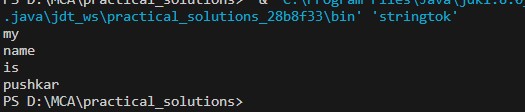
System.out.println(st.nextToken());

}

}

}

**Output:**



**Q4. Write a program to demonstrate user defined exception.**

import java.io.\*; import java.util.\*;

class namenotvalid extends Exception

{

namenotvalid(String msg)

{

super(msg);

}

}

class throw4 {

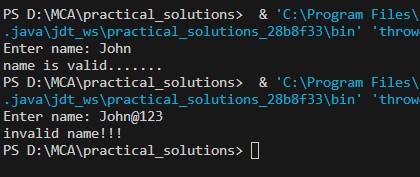
public static void main(String[] args) {

// BufferedReader br=new BufferedReader(new

InputStreamReader(System.in));

Scanner sc=new Scanner(System.in);

|  |  |  |
| --- | --- | --- |
| //                                      }  **Output:** | } | try {  System.out.print("Enter name: ");  String name=br.readLine(); String name=sc.next(); for(int i=0;i<name.length();i++)  {  char x=name.charAt(i);  if(x=='@' || x=='%' || x=='$' || x=='#' || x=='\*' || x=='&') throw new namenotvalid("invalid name!!!");  }  System.out.println("name is valid.......");  }  catch(namenotvalid e)  {  System.out.println(e.getMessage());  }  catch(Exception ed)  {  ed.printStackTrace();  }  15 |



**Q5. Write a program to create a thread using a) Extending the Thread class**

**b) Implementing Runnable interface.**

a)

class thread extends Thread

{

String msg; thread(String message)

{

this.msg=message;

} public void run()

{ try { for(int i=1;i<=5;i++)

{

System.out.println(msg+"-"+i);

Thread.sleep(5000);

} }

catch(InterruptedException e)

{

}

}

}

public class MyThread

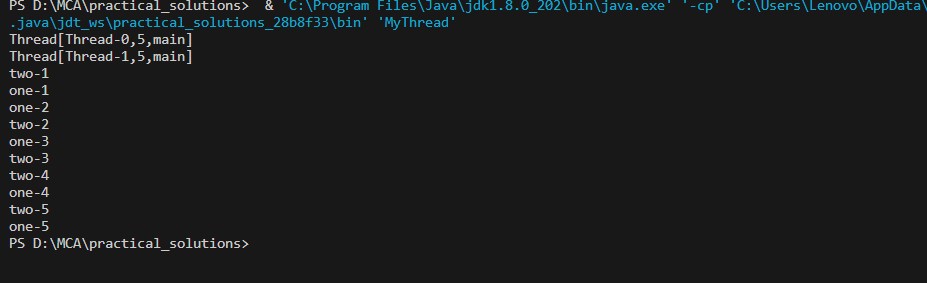
{

public static void main(String[] args) { thread t1=new thread("one"); thread t2=new thread("two"); System.out.println(t1); System.out.println(t2); t1.start(); t2.start();

}

}

**Output:**



b)

class MyRunnable implements Runnable { public void run() {

for (int i = 1; i <= 5; i++) {

System.out.println(Thread.currentThread().getId() + " Value " + i);

}

}

}

public class RunnableExample { public static void main(String args[]) {

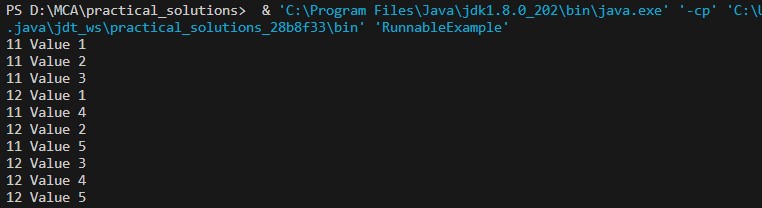
Thread t1 = new Thread(new MyRunnable());

Thread t2 = new Thread(new MyRunnable()); t1.start(); t2.start();

}

}

**Output:**



**Q6. Write a program to copy the contents of one file into another file in reverse direction.**

import java.io.RandomAccessFile;

public class reverse {

public static void main(String[] args) {

String sourceFilePath = "file1.txt";

String destinationFilePath = "reversed\_file.txt";

try {

reverseCopyFile(sourceFilePath, destinationFilePath);

System.out.println("File copied successfully in reverse order.");

} catch (Exception e) {

System.err.println("Error copying file: " + e.getMessage());

}

}

private static void reverseCopyFile(String source, String destination) throws Exception

{

RandomAccessFile sourceFile = new RandomAccessFile(source, "r");

RandomAccessFile destinationFile = new RandomAccessFile(destination,

"rw");

long sourceLength = sourceFile.length();

// Start reading from the end of the source file

for (long pointer = sourceLength - 1; pointer >= 0; pointer--) { sourceFile.seek(pointer); char c = (char) sourceFile.read(); destinationFile.write(c);

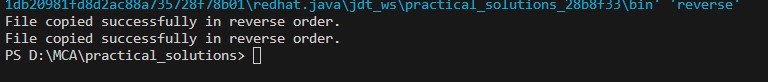
}

System.out.println("File copied successfully in reverse order.");

}

}

**Output:**



**Q7. Write a program to display the contents of a file.**

import java.util.Scanner; import java.io.\*;

public class fileread

{

public static void main(String[] args)

{

String fname;

Scanner sc = new Scanner(System.in);

// enter filename along with its extension

System.out.print("Enter the Name of File: "); fname = sc.nextLine();

String line = null; try

{

FileReader fileReader = new FileReader(fname);

// always wrap the FileReader in BufferedReader

BufferedReader br = new BufferedReader(fileReader);

while((line = br.readLine()) != null)

{

System.out.println(line);

}

// always close the file after its use br.close(); }

catch(IOException ex)

{

System.out.println("\nError occurred");

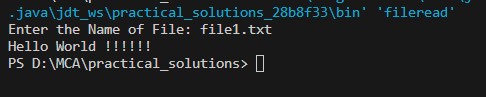
System.out.println("Exception Name: " +ex);

}

}

}

**Output:**



**Q8. Write a program to Create Calculator by using AWT.**

import java.awt.\*; import java.awt.event.\*;

public class calculator implements ActionListener{ int c,n;

String s1,s2,s3,s4,s5;

Frame f;

Button b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, badd, bsub, bmul, bdiv, beq, bclr;

Panel p;

TextField t1;

GridLayout g;

calculator(){ f = new Frame("Calculator");

f.setLayout(new FlowLayout()); p = new Panel(); b0 = new Button("0"); b0.addActionListener(this);

b1 = new Button("1"); b1.addActionListener(this);

b2 = new Button("2"); b2.addActionListener(this);

b3 = new Button("3"); b3.addActionListener(this);

b4 = new Button("4"); b4.addActionListener(this);

b5 = new Button("5"); b5.addActionListener(this);

b6 = new Button("6"); b6.addActionListener(this);

b7 = new Button("7"); b7.addActionListener(this);

b8 = new Button("8"); b8.addActionListener(this);

b9 = new Button("9"); b9.addActionListener(this);

badd = new Button("+"); badd.addActionListener(this);

bsub = new Button("-"); bsub.addActionListener(this);

bmul = new Button("\*"); bmul.addActionListener(this);

bdiv = new Button("/"); bdiv.addActionListener(this); beq = new Button("="); beq.addActionListener(this);

bclr = new Button("CLR"); bclr.addActionListener(this);

t1 = new TextField(20); f.add(t1); g = new GridLayout(4,4); p.setLayout(g);

p.add(b0);

p.add(b1);

p.add(b2);

p.add(b3);

p.add(b4);

p.add(b5);

p.add(b6);

p.add(b7);

p.add(b8);

p.add(b9);

p.add(badd);

p.add(bsub);

p.add(bmul);

p.add(bdiv);

p.add(beq);

p.add(bclr);

f.add(p);

f.setSize(300,300);

f.setVisible(true);

f.setBackground(Color.LIGHT\_GRAY);

f.addWindowListener(new WindowAdapter() { @Override

public void windowClosing(WindowEvent e) {

System.exit(0);

}

});

}

@Override

public void actionPerformed(ActionEvent e) { if(e.getSource()==b0){

s3 = t1.getText(); s4 = "0"; s5 = s3 + s4; t1.setText(s5); } if(e.getSource()==b1){ s3 = t1.getText(); s4 = "1"; s5 = s3 + s4; t1.setText(s5); } if(e.getSource()==b2){ s3 = t1.getText(); s4 = "2"; s5 = s3 + s4; t1.setText(s5); } if(e.getSource()==b3){ s3 = t1.getText(); s4 = "3"; s5 = s3 + s4; t1.setText(s5);

} if(e.getSource()==b4){ s3 = t1.getText(); s4 = "4"; s5 = s3 + s4; t1.setText(s5); } if(e.getSource()==b5){ s3 = t1.getText(); s4 = "5"; s5 = s3 + s4; t1.setText(s5); } if(e.getSource()==b6){ s3 = t1.getText(); s4 = "6"; s5 = s3 + s4; t1.setText(s5); } if(e.getSource()==b7){ s3 = t1.getText(); s4 = "7"; s5 = s3 + s4; t1.setText(s5);

} if(e.getSource()==b8){ s3 = t1.getText(); s4 = "8"; s5 = s3 + s4; t1.setText(s5); } if(e.getSource()==b9){ s3 = t1.getText(); s4 = "9"; s5 = s3 + s4; t1.setText(s5); } if(e.getSource()==badd){ s1 = t1.getText(); t1.setText(""); c = 1; } if(e.getSource()==bsub){ s1 = t1.getText(); t1.setText(""); c = 2; } if(e.getSource()==bmul){ s1 = t1.getText(); t1.setText(""); c = 3; }

if(e.getSource()==bdiv){ s1 = t1.getText(); t1.setText(""); c = 4; }

if(e.getSource()==beq){ s2 = t1.getText(); if(c==1){

n = Integer.parseInt(s1) + Integer.parseInt(s2); t1.setText(String.valueOf(n));

} if(c==2){

n = Integer.parseInt(s1) - Integer.parseInt(s2); t1.setText(String.valueOf(n));

} if(c==3){

n = Integer.parseInt(s1) \* Integer.parseInt(s2); t1.setText(String.valueOf(n));

} if(c==4){

n = Integer.parseInt(s1) / Integer.parseInt(s2); t1.setText(String.valueOf(n));

} }

if(e.getSource()==bclr){ t1.setText("");

}

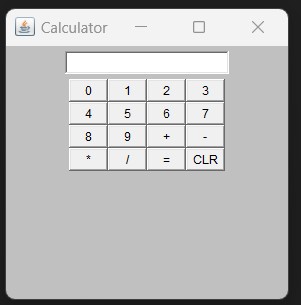
}

public static void main(String[] args) { calculator c = new calculator();

}

}

**Output:**



**Q9. Write a program to create a menu using AWT / Swing.**

import javax.swing.\*; import java.awt.event.\*;

public class MenuExample implements ActionListener {

JFrame f;

JMenuBar mb;

JMenu file,edit,help;

JMenuItem cut,copy,paste,selectAll;

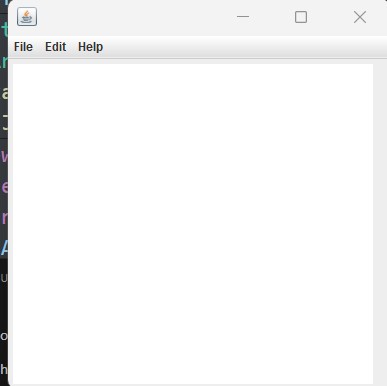
JTextArea ta; MenuExample(){ f=new JFrame(); cut=new JMenuItem("cut"); copy=new JMenuItem("copy"); paste=new JMenuItem("paste"); selectAll=new JMenuItem("selectAll"); cut.addActionListener(this); copy.addActionListener(this); paste.addActionListener(this); selectAll.addActionListener(this); mb=new JMenuBar(); file=new JMenu("File"); edit=new JMenu("Edit"); help=new JMenu("Help"); edit.add(cut); edit.add(copy); edit.add(paste); edit.add(selectAll); mb.add(file); mb.add(edit); mb.add(help); ta=new JTextArea(); ta.setBounds(5,5,360,320);

f.add(mb);

f.add(ta);

|  |  |
| --- | --- |
| } | f.setJMenuBar(mb);  f.setLayout(null);  f.setSize(400,400);  f.setVisible(true); }  public void actionPerformed(ActionEvent e) { if(e.getSource()==cut) ta.cut();  if(e.getSource()==paste) ta.paste();  if(e.getSource()==copy) ta.copy();  if(e.getSource()==selectAll) ta.selectAll();  }  public static void main(String[] args) { new MenuExample();  }  35 |

**Output:**



**Q10. Write a program to Create Log in form by using AWT/Swing and JDBC.**

import javax.swing.\*; import java.awt.\*;

import java.awt.event.ActionEvent; import java.awt.event.ActionListener;

public class LoginFrame extends JFrame implements ActionListener {

Container container = getContentPane();

JLabel userLabel = new JLabel("USERNAME");

JLabel passwordLabel = new JLabel("PASSWORD");

JTextField userTextField = new JTextField();

JPasswordField passwordField = new JPasswordField();

JButton loginButton = new JButton("LOGIN");

JButton resetButton = new JButton("RESET");

JCheckBox showPassword = new JCheckBox("Show Password");

LoginFrame() { setLayoutManager(); setLocationAndSize(); addComponentsToContainer(); addActionEvent();

}

public void setLayoutManager() { container.setLayout(null);

}

public void setLocationAndSize() { userLabel.setBounds(50, 150, 100, 30); passwordLabel.setBounds(50, 220, 100, 30); userTextField.setBounds(150, 150, 150, 30); passwordField.setBounds(150, 220, 150, 30); showPassword.setBounds(150, 250, 150, 30); loginButton.setBounds(50, 300, 100, 30); resetButton.setBounds(200, 300, 100, 30);

}

public void addComponentsToContainer() { container.add(userLabel); container.add(passwordLabel); container.add(userTextField); container.add(passwordField); container.add(showPassword); container.add(loginButton); container.add(resetButton);

}

public void addActionEvent() { loginButton.addActionListener(this); resetButton.addActionListener(this); showPassword.addActionListener(this);

}

@Override

public void actionPerformed(ActionEvent e) { //Coding Part of LOGIN button if (e.getSource() == loginButton) {

String userText; String pwdText;

userText = userTextField.getText(); pwdText = passwordField.getText(); if (userText.equalsIgnoreCase("mehtab") && pwdText.equalsIgnoreCase("12345")) {

JOptionPane.showMessageDialog(this, "Login Successful");

} else {

JOptionPane.showMessageDialog(this, "Invalid Username or Password");

}

}

//Coding Part of RESET button if (e.getSource() == resetButton) { userTextField.setText(""); passwordField.setText("");

}

//Coding Part of showPassword JCheckBox if (e.getSource() == showPassword) { if (showPassword.isSelected()) { passwordField.setEchoChar((char) 0);

} else {

passwordField.setEchoChar('\*');

}

} }

public static void main(String[] a) {

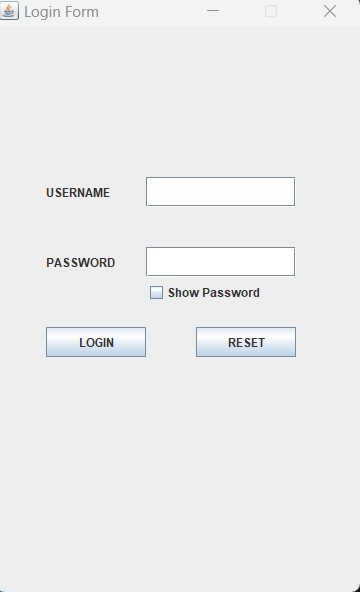
LoginFrame frame = new LoginFrame(); frame.setTitle("Login Form"); frame.setVisible(true); frame.setBounds(10, 10, 370, 600);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); frame.setResizable(false);

}

}

**Output:**



**Q11. Write a program to demonstrate operations performed on ArrayList.**

import java.util.ArrayList; import java.util.Iterator;

public class ArrayListOperations { public static void main(String[] args) {

// Creating an ArrayList of integers

ArrayList<Integer> numbersList = new ArrayList<>();

// Adding elements to the ArrayList numbersList.add(10); numbersList.add(20); numbersList.add(30); numbersList.add(40);

// Displaying the original ArrayList

System.out.println("Original ArrayList: " + numbersList);

// Accessing elements using get() method int elementAtIndex2 = numbersList.get(2);

System.out.println("Element at index 2: " + elementAtIndex2);

// Updating an element at a specific index numbersList.set(1, 25);

System.out.println("ArrayList after updating element at index 1: " + numbersList);

// Removing an element by value numbersList.remove(Integer.valueOf(30));

System.out.println("ArrayList after removing value 30: " + numbersList); // Iterating through the ArrayList using an iterator

System.out.print("ArrayList elements using iterator: "); Iterator<Integer> iterator = numbersList.iterator(); while (iterator.hasNext()) {

System.out.print(iterator.next() + " ");

}

System.out.println();

// Checking if an element exists in the ArrayList boolean containsElement = numbersList.contains(40);

System.out.println("Does ArrayList contain 40? " + containsElement);

// Getting the size of the ArrayList int sizeOfArrayList = numbersList.size();

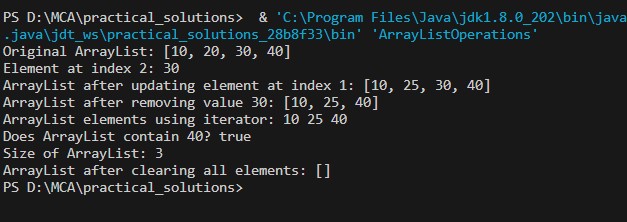
System.out.println("Size of ArrayList: " + sizeOfArrayList);

// Clearing all elements from the ArrayList numbersList.clear();

System.out.println("ArrayList after clearing all elements: " + numbersList); }

}

**Output:**



**Q12. Write a program to demonstrate operations performed on LinkedList.**

import java.util.LinkedList; import java.util.Iterator;

public class LinkedListOperations { public static void main(String[] args) {

// Creating a LinkedList of strings

LinkedList<String> colorsList = new LinkedList<>();

// Adding elements to the LinkedList colorsList.add("Red"); colorsList.add("Green"); colorsList.add("Blue"); colorsList.add("Yellow");

// Displaying the original LinkedList

System.out.println("Original LinkedList: " + colorsList);

// Adding elements at specific positions colorsList.add(2, "Purple"); colorsList.addFirst("Black"); colorsList.addLast("White");

System.out.println("LinkedList after adding elements: " + colorsList);

// Accessing elements using get() method

String elementAtIndex3 = colorsList.get(3);

System.out.println("Element at index 3: " + elementAtIndex3);

// Updating an element at a specific index colorsList.set(1, "Brown");

System.out.println("LinkedList after updating element at index 1: " + colorsList);

// Removing an element by value colorsList.remove("Blue");

System.out.println("LinkedList after removing value 'Blue': " + colorsList); // Iterating through the LinkedList using an iterator

System.out.print("LinkedList elements using iterator: "); Iterator<String> iterator = colorsList.iterator(); while (iterator.hasNext()) {

System.out.print(iterator.next() + " ");

}

System.out.println();

// Checking if an element exists in the LinkedList boolean containsElement = colorsList.contains("Green");

System.out.println("Does LinkedList contain 'Green'? " + containsElement);

// Getting the size of the LinkedList int sizeOfLinkedList = colorsList.size();

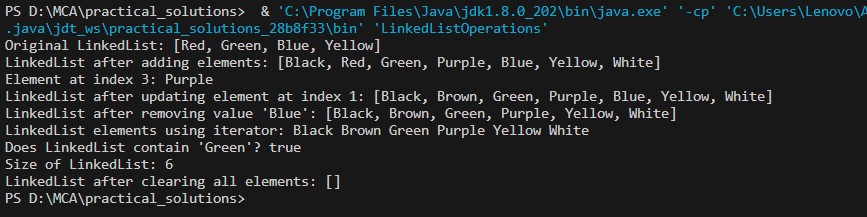
System.out.println("Size of LinkedList: " + sizeOfLinkedList);

// Clearing all elements from the LinkedList colorsList.clear();

System.out.println("LinkedList after clearing all elements: " + colorsList); }

}

**Output:**



**Q13. Write a JDBC program to Perform CRUD Operations on Oracle Database. (4 Different Programs)**

**A) Create(Insert)**

import java.sql.\*;

class AccountStoringApplication

{

public static void main(String[] args) throws ClassNotFoundException, SQLException

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con =

DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","System",

"pranaya");

Statement st = con.createStatement();

int c = st.executeUpdate("insert into account values(1005, 'pranaya', 2345)"); System.out.println(c + "account stored successfully");

int c = st.executeUpdate("insert into account values(1006, 'kumar', 5345)"); System.out.println(c + "more account stored successfully"); st.close(); con.close();

}

}

//CREATE TABLE ACCOUNT (accnonumber(8) primary key, name varchar2(12), balance number(8,2));

**B)Update**

import java.sql.\*;

class UpdateAccountApplication

{

public static void main (String[]args) throws ClassNotFoundException, SQLException

{

Class.forName ("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe", "System", "pranaya");

Statement st = con.createStatement ();

int rows = st.executeUpdate ("update account set balance = balance+2000"); System.out.println (rows + " rows modified"); st.close (); con.close ();

}

}

**C)Read(Select)**

import java.sql.\*; import java.util.\*; class AccountDetails

{

public static void main (String[]args) throws ClassNotFoundException, SQLException

{

Scanner sc = new Scanner (System.in);

System.out.println ("ENTER ACCOUNT NUMBER"); int ano = sc.nextInt ();

Class.forName ("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe", "System", "pranaya");

Statement st = con.createStatement ();

ResultSet rs = st.executeQuery ("select \* from account where accno =" + ano); if (rs.next ())

{

System.out.println ("account no: " + rs.getInt (1));

System.out.println ("acc holder name:" + rs.getString (2));

System.out.println ("balance : " + rs.getFloat (3));

System.out.println ("address: " + rs.getString (4));

} else

System.out.println ("account doesnot exist");

rs.close (); st.close (); con.close ();

}

}

**D)Delete**

import java.sql.\*; import java.util.\*;

class AccountCloseApplication

{

public static void main (String[]args) throws ClassNotFoundException, SQLException

{

Scanner sc = new Scanner (System.in);

System.out.println ("ENTER ACCOUNT NUMBER"); int ano = sc.nextInt ();

Class.forName ("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection

("jdbc:oracle:thin:@localhost:1521:xe", "System", "pranaya"); Statement st = con.createStatement ();

int c = st.executeUpdate ("delete from account where accno =" + ano); if (c == 0)

System.out.println ("account doesnot exist"); else

System.out.println ("account closed successfully"); st.close (); con.close ();

}

}

**Q14. Write a program to Connect Java Application with MySQL Database.**

public class MySQLConnectionDemo {

// JDBC URL, username, and password of MySQL server

static final String JDBC\_URL = "jdbc:mysql://localhost:3306/your\_database\_name"; static final String USERNAME = "your\_username"; static final String PASSWORD = "your\_password";

public static void main(String[] args) { Connection connection = null;

Statement statement = null;

ResultSet resultSet = null;

try {

// Register JDBC driver

Class.forName("com.mysql.cj.jdbc.Driver");

// Open a connection

System.out.println("Connecting to database...");

connection = DriverManager.getConnection(JDBC\_URL, USERNAME, PASSWORD);

// Execute a query

statement = connection.createStatement();

String sqlQuery = "SELECT \* FROM your\_table\_name"; resultSet = statement.executeQuery(sqlQuery);

// Process the result set while (resultSet.next()) {

// Assuming a table with columns 'column1', 'column2' int column1Value = resultSet.getInt("column1");

String column2Value = resultSet.getString("column2");

// Print or process the retrieved data

System.out.println("Column1: " + column1Value + ", Column2: " + column2Value); }

} catch (SQLException | ClassNotFoundException e) { e.printStackTrace();

} finally {

// Close resources in reverse order of their creation try {

if (resultSet != null) resultSet.close(); if (statement != null) statement.close(); if (connection != null) connection.close(); } catch (SQLException e) { e.printStackTrace();

}

}

}

}

**Q15. Write a servlet program to implement Get and Post methods.**

**Index.html**

<html>

<body>

<form method="post" action="servlet3.java" >

User name <input type="text" name="uname" />

Password <input type="password" name="pwd" />

<input type="submit" value="Login" />

</form>

</body>

</html>

**Servlet3.java**

import java.io.\*; import javax.servlet.\*; import javax.servlet.http.\*; public class servlet3 extends HttpServlet

{

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

{

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String u = request.getParameter("uname");

String p = request.getParameter("pwd"); String valid = null;

if ((u.equals("admin")) && (p.equals("rose"))) valid = "Successful"; else

valid = "Unsuccessful"; out.println("<html>"); out.println("<body>");

out.println("<h1> Your authentication is " + valid + "</h1>"); out.println("</body></html>");

}

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException { doPost(request, response);

}

}

**Q16. Write a JSP program.**

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>Simple JSP Program</title>

</head>

<body>

<h1>Hello, World! This is a simple JSP program.</h1>

</body>

</html>

**Output :**



**INDEX**

# Data Structures and Algorithms (IT11L)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.No** | **Date** | **Title** | **Page.no** | **Teacher’s**  **sign** |
| 1. | 14/09/2023 | Write a JS program for insertion operations on Singly LinkedList |  |  |
| 2. | 14/09/2023 | Write a JS program for insertion operations on Doubly LinkedList |  |  |
| 3. | 20/09/2023 | Write a JS program to create a singly linked list and count total number of nodes in it and  display the result |  |  |
| 4. | 20/09/2023 | Write a JS program for stack with array implementation |  |  |
| 5. | 21/09/2023 | Write a JS program for array implementation of circular Queue for integers |  |  |
| 6. | 21/09/2023 | Write a JS program to reverse a string using stack |  |  |
| 7. | 27/09/2023 | Write a JS program for Doubly linked list-Sort the linked list in ascending order.  And display it |  |  |
| 8. | 27/09/2023 | Write a JS program for Graph implementation and DFS graph traversals |  |  |
| 9. | 4/10/2023 | Write a JS Program to print BFS traversal from a given source vertex |  |  |
| 10. | 4/10/2023 | Write a JS program to Implement Min Heap |  |  |
| 11. | 5/10/2023 | Write a JS program to Implement Max Heap |  |  |
| 12. | 5/10/2023 | Write a JS program for implementation of  Hashing |  |  |
| 13. | 18/10/2023 | Write a JS program Rain water Trapping (Practical based on Brute Force technique) |  |  |
| 14. | 18/10/2023 | Write a JS program Jump Game.( Practical based on Greedy Algorithm) |  |  |
| 15. | 19/10/2023 | Write a JS program for Binary  Search(practical based on Divide and Conquer technique) |  |  |
| 16. | 19/10/2023 | Write a JS program for finding out power set(practical based on Backtracking) |  |  |
| 17. | 25/10/2023 | Write a JS program for BST |  |  |
| 18. | 25/10/2023 | Write a JS program for compute a^n where n is positive integer using fast power method |  |  |
| 19. | 26/10/2023 | Write a JS program for finding GCD using  Euclidean algorithm |  |  |
| 20. | 26/10/2023 | Write a JS program to create a Pascal’s  Triangle |  |  |
| 21. | 1/11/2023 | Write a JS program Dijkstra shortest path algorithm using Prim’s Algorithm |  |  |
| 22. | 1/11/2023 | Write JS program for sorting array using quick sort |  |  |
| 23. | 2/11/2023 | Write JS program for staircase problem |  |  |
| 24. | 2/11/2023 | Write JS program for tower of hanoi |  |  |
| 25. | 8/11/2023 | Write JS program for powerset |  |  |
| 26. | 8/11/2023 | Write JS program for binarysearch |  |  |
| 27. | 9/11/2023 | Write JS program for Euclidian algorithm |  |  |
| 28. | 9/11/2023 | Write JS program for fastpowering algorithm |  |  |

**Q1.** **1.Write a JS program for insertion operations on Singly LinkedList Insert At Front**

var head; class Node

{

constructor(val)

{

this.data=val; this.next=null;

}

}

//front of list function push(new\_data)

{

var new\_node=new Node(new\_data); new\_node.next=head; head=new\_node;

}

//at given position

function insertAfter(prev\_node,new\_data)

{

if(prev\_node==null)

{

console.log("The given previous noe cannot be null"); return; }

var new\_node=new Node(new\_data); new\_node.next=prev\_node.next; prev\_node.next=new\_node;

}

//appends at end function append(new\_data)

{

var new\_node=new Node(new\_data); if(head==null) {

head=new Node(new\_data); return; } new\_node.next=null; var last=head; while(last.next!=null) last=last.next;

last.next=new\_node; return;

}

function printList()

{

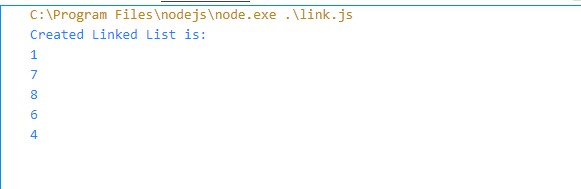
var tnode=head; while(tnode!=null) { console.log(tnode.data+" "); tnode=tnode.next;

}

}

append(6); push(7); push(1); append(4);

insertAfter(head.next,8); console.log("Created Linked List is:"); printList(); **output :**



**2.insert mid:** var head; class Node {

//constructor to create a new node constructor(val) { this.data = val; this.next = null;

}

}

//function to insert node at the middle of the linked list function insertAtMid(x) { //if list is empty if (head == null) head = new Node(x); else {

var newNode = new Node(x);

var slow = head; var fast = head.next;

While(fast != null && fast.next != null)

{ slow=slow.next; fast=fast.net.next; } newNode.next = slow.next; slow.next = newNode;

}

}

//function to display the linke list function display() { var temp = head; while (temp != null) { console.log(temp.data + " "); temp = temp.next;

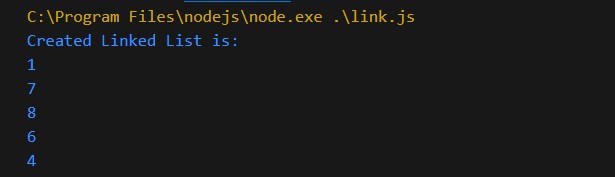
}

}

head = null; head = new Node(1); head.next = new Node(2); head.next.next = new Node(4); head.next.next.next = new Node(5); console.log("linked list after" + "insertion:"); display(); var x = 3; insertAtMid(x);

console.log("<br/>Linked list after" + "insertion:"); display();

**output :**



**Q2.****Write a JS program for insertion operations on Doubly LinkedList**

1. **Insert At Front**
2. **Insert At Middle**

var head; class Node { constructor(d) { this.data = d; this.next = null; this.prev = null;

}

}

// Adding a node at the front of the list function push(new\_data) {

var new\_Node = new Node(new\_data); new\_Node.next = head; new\_Node.prev = null;

if (head != null) head.prev = new\_Node;

head = new\_Node;

}

// Add a node before the given node function InsertBefore(next\_node , new\_data) {

if (next\_node == null) { console.log("The given next node can not be NULL"); return;

}

var new\_node = new Node(new\_data); new\_node.prev = next\_node.prev; next\_node.prev = new\_node; new\_node.next = next\_node;

if (new\_node.prev != null) new\_node.prev.next = new\_node; else head = new\_node;

}

function InsertAfter(prev\_Node , new\_data) {

if (prev\_Node == null) { console.log("The given previous node cannot be NULL "); return;

}

var new\_node = new Node(new\_data); new\_node.next = prev\_Node.next; prev\_Node.next = new\_node; new\_node.prev = prev\_Node;

if (new\_node.next != null) new\_node.next.prev = new\_node;

}

function append(new\_data) {

var new\_node = new Node(new\_data); var last = head; new\_node.next = null;

if (head == null) { new\_node.prev = null; head = new\_node; return;

}

while (last.next != null) last = last.next;

last.next = new\_node; new\_node.prev = last;

} function printlist(node) { var last = null; console.log("<br/>Traversal in forward Direction<br/>"); while (node != null) { console.log(node.data + " "); last = node; node = node.next;

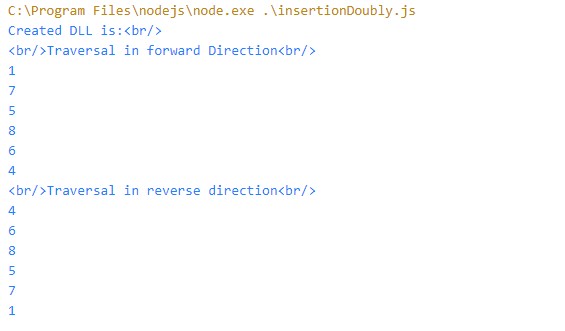
} console.log(); console.log("<br/>Traversal in reverse direction<br/>"); while (last != null) { console.log(last.data + " "); last = last.prev;

}

} append(6); push(7); push(1); append(4);

InsertAfter(head.next, 8); InsertBefore(head.next.next, 5); console.log("Created DLL is:<br/> "); printlist(head);

**output :**



**Q3.****. Write a JS program to create a singly linked list and count total number of nodes in it and display the result.**

class Node

{

constructor(data)

{ this.data=data; this.next=null;

}

}

class DoublyLinkedList

{

constructor()

{ this.head=null; this.size=0; } insertFirst(data) { var node=new Node(data); if(this.head===null) { this.head=node; } else { node.next=this.head; this.head=node; } this.size++; } countNodes() { let count=0; let current=this.head; while(current) { count++; current=current.next;

} console.log("The total number of nodes:"+count);

} printList() { var current=this.head; while(current) { console.log(current.data); current=current.next;

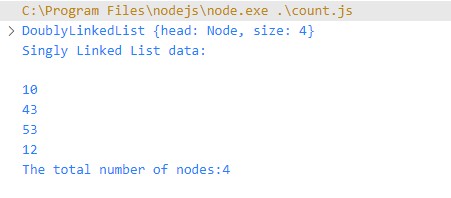
}

}

}

var dll=new DoublyLinkedList(); dll.insertFirst(12); dll.insertFirst(53); dll.insertFirst(43); dll.insertFirst(10); console.log(dll); console.log("Singly Linked List data:\n"); dll.printList(); dll.countNodes();

**output :**



**Q4.****Write a JS program for stack with array implementation-**

1. **To check is empty.**
2. **To Peek.**
3. **To PUSH.**
4. **and POP the stack.**

class Stack

{

constructor()

{ this.item=[];

} push(data) { this.item.push(data);

} pop() { if(this.item.length==0) return "underflow"; return this.item.pop();

} peek() { return this.item[this.item.length-1];

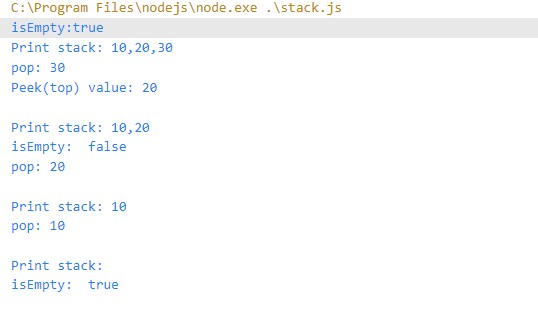
} isEmpty() { return this.item.length==0;

}

}

var s1=new Stack(); console.log("isEmpty:"+s1.isEmpty()); s1.push(10); s1.push(20); s1.push(30); console.log("Print stack: "+s1.item); console.log("pop: "+s1.pop()); console.log("Peek(top) value: "+s1.peek()); console.log("\nPrint stack: "+s1.item); console.log("isEmpty: "+s1.isEmpty()); console.log("pop: "+s1.pop()); console.log("\nPrint stack: "+s1.item); console.log("pop: "+s1.pop()); console.log("\nPrint stack: "+s1.item); console.log("isEmpty: "+s1.isEmpty());

**output :**



**Q5.****Write a JS program for array implementation of circular Queue for integers- 1. Insert.**

1. **Delete.**
2. **Display.**

class CirculerQueue

{

constructor(size)

{ this.data=[]; this.size=size; this.length=0; this.front=0; this.rear=-1;

} isEmpty() { return (this.length==0)

} enqueue(element)

{ if(this.length>=this.size) console.log("full"); this.rear++; this.data[(this.rear)%this.size]=element;// data[0]=10 this.length++;

} getfront() { if(this.length==0)

{ console.log("no element in circular queue");

}

return this.data[this.front%this.size]

} dequeue() { if(this.length==0) console.log("no element");

const value=this.getfront(); this.data[this.front%this.size]=null; this.front++; this.length--; console.log("dequeue: "+value);

} printQueue() { var str=""; for(var i=this.front; i!==(this.rear)+1; i++)

{ str+=this.data[i]+" ";

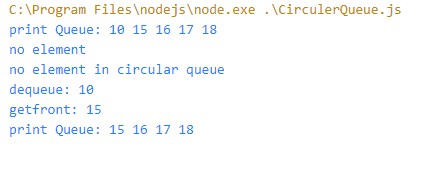
} return str;

}

}

var cq=new CirculerQueue(5); cq.enqueue(10); cq.enqueue(15); cq.enqueue(16); cq.enqueue(17); cq.enqueue(18); console.log("print Queue: "+cq.printQueue()); cq.dequeue(); console.log("getfront: "+cq.getfront()); console.log("print Queue: "+cq.printQueue());

**output :**



**Q6.** **Write JS program to reverse a string using stack.**

class revStack

{ reverse(str) { let stack=[]; let reverseStr; for(let i=0;i<str.length;i++)

{ stack.push(str[i]);

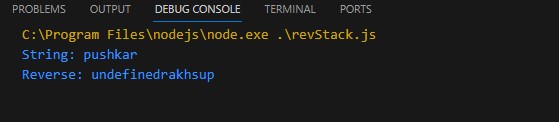
} while(stack.length>0)

{ reverseStr+=stack.pop();

} return reverseStr;

} } var s1=new revStack(); console.log("String: pushkar \nReverse: "+s1.reverse("pushkar"));

**output :**



**Q7. Write a JS program for Doubly linked list-Sort the linked list in ascending order. And display it.**

class Node {

constructor(data) { this.data = data; this.prev = null; this.next = null;

}

} class DoublyLinkedList { constructor() { this.head = null; this.tail = null;

} append(data) { const newNode = new Node(data);

if (!this.head) { this.head = newNode; this.tail = newNode; } else { newNode.prev = this.tail; this.tail.next = newNode; this.tail = newNode;

}

}

display() { let current = this.head; while (current) { console.log(current.data); current = current.next;

}

} sort() { let current = this.head;

while (current) { let temp = current.next;

while (temp) { if (current.data > temp.data) { // Swap data const tempData = current.data; current.data = temp.data; temp.data = tempData;

} temp = temp.next;

}

current = current.next;

}

}

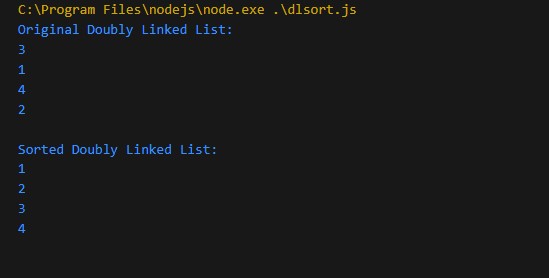
}

// Example usage: const doublyLinkedList = new DoublyLinkedList(); doublyLinkedList.append(3); doublyLinkedList.append(1); doublyLinkedList.append(4); doublyLinkedList.append(2);

console.log("Original Doubly Linked List:"); doublyLinkedList.display();

console.log("\nSorted Doubly Linked List:"); doublyLinkedList.sort(); doublyLinkedList.display();

**output :**



**Q8. Write a JS program for Graph implementation and DFS graph traversals.**

class Graph{ constructor(v){ this.V=v; this.adj=new Array(v); for(let i=0;i<v;i++) this.adj[i]=[];

} addEdge(v,w){ this.adj[v].push(w);

}

DFSUtil(v,visited){ visited[v]=true; console.log(v+" "); for(let i of this.adj[v].values()) { let n=i; if(!visited[n]) this.DFSUtil(n,visited);

}

}

DFS(v) { let visited=new Array(this.v); for(let i=0;i<this.V;i++) visited[i]=false; this.DFSUtil(v,visited);

}

} g=new Graph(4);

g.addEdge(0,1);

g.addEdge(0,2);

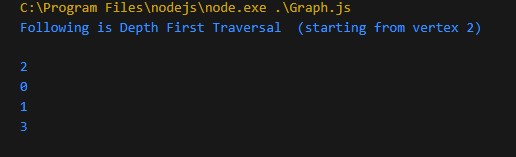
g.addEdge(1,2);

g.addEdge(2,0);

g.addEdge(2,3);

g.addEdge(3,3); console.log("Following is Depth First Traversal "+"(starting from vertex 2)\n"); g.DFS(2);

**output :**



**Q9. Write JS Program to print BFS traversal from a given source vertex.**

class BFS { constructor(v){ this.V=v; this.adj=new Array(v); for(let i=0;i<v;i++) this.adj[i]=[];

} addEdge(v,w)

{ this.adj[v].push(w);

} BFS(s){ let visited=new Array(this.V); for(let i=0;i<this.V;i++) visited[i]=false;

let queue=[]; visited[s]=true; queue.push(s); while(queue.length>0)

{ s=queue[0]; console.log(s+" "); queue.shift(); this.adj[s].forEach((adjacent,i) =>{ if(!visited[adjacent]) { visited[adjacent]=true; queue.push(adjacent);

}

});

}

}

}

g=new BFS(4);

g.addEdge(0,1);

g.addEdge(0,2);

g.addEdge(1,2);

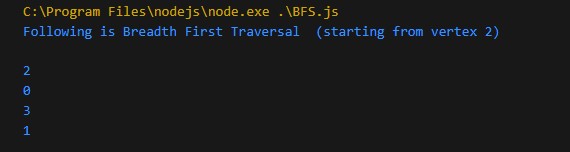
g.addEdge(2,0);

g.addEdge(2,3);

g.addEdge(3,3);

console.log("Following is Breadth First Traversal "+"(starting from vertex 2)\n"); g.BFS(2);

**output:**



**Q10. Write JS program to Implement Min Heap.**

class minHeap { constructor() { this.heap = []; this.elements = 0;

}; insert(val) { if (this.elements >= this.heap.length) { this.elements = this.elements + 1; this.heap.push(val); this.\_percolateUp(this.heap.length - 1);

} else

{ this.heap[this.elements] = val; this.elements = this.elements + 1; this.\_percolateUp(this.heap.length - 1);

}

};

getMin(){ if (this.heap.length !== 0) return this.heap[0];

//return null

}

removeMin()

{ const min = this.heap[0]; if (this.elements > 1) { this.heap[0] = this.heap[this.elements - 1]; this.elements = this.elements - 1; this.\_minHeapify(0); return min;

} else if (this.elements == 1) { this.elements = this.elements - 1; return min; } else {

return null;

}

};

\_percolateUp(index)

{ let parent = Math.floor((index - 1) / 2); if (index <= 0) return else if (this.heap[parent] > this.heap[index]) { let tmp = this.heap[parent]; this.heap[parent] = this.heap[index]; this.heap[index] = tmp; this.\_percolateUp(parent);

}

};

\_minHeapify(index){

let left = (index \* 2) + 1; let right = (index \* 2) + 2; let smallest = index; if ((this.elements > left) && (this.heap[smallest] > this.heap[left])) { smallest = left;

} if ((this.elements > right) && (this.heap[smallest] > this.heap[right])) smallest = right; if (smallest !== index) { let tmp = this.heap[smallest]; this.heap[smallest] = this.heap[index]; this.heap[index] = tmp; this.\_minHeapify(smallest);

}

}

buildHeap(arr){ this.heap = arr; this.elements = this.heap.length; for (let i = this.heap.length - 1; i >= 0; i--) { this.\_minHeapify(i);

}

}

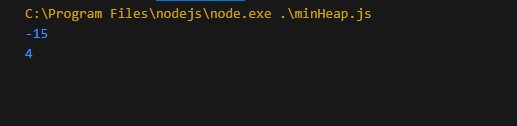
};

let heap = new minHeap(); heap.insert(12); heap.insert(10); heap.insert(-10); heap.insert(100); heap.insert(-15); console.log(heap.getMin()); let newheap = new minHeap();

arr = [12, 6, 8, 3, 16, 4, 27]; newheap.buildHeap(arr);

newheap.removeMin(); console.log(newheap.getMin());

**output :**



**Q11. Write JS program to Implement Max Heap .**

class maxHeap{ constructor() { this.heap=[]; this.elements=0;

}; insert(val) { if(this.elements>=this.heap.length)

{ this.elements=this.elements+1; this.heap.push(val); this.\_percolateUp(this.heap.length-1);

} else { this.heap[this.elements]=val; this.elements=this.elements+1; this.\_percolateUp(this.elements-1);

} }; getMax() { if(this.elements!==0) return this.heap[0]; return null;

}; removeMax() { let max=this.heap[0]; if(this.elements>1)

{

this.heap[0]=this.heap[this.elements-1]; this.elements=this.elements-1; this.\_maxHeapify(0); return max; } else if(this.elements ===1)

{ this.elements=this.elements-1; return max; } else{ return null;

}

};

\_percolateUp(index)

{ const parent=Math.floor((index-1)/2);

if(index<=0) return else if(this.heap[parent]<this.heap[index])

{ let tmp=this.heap[parent]; this.heap[parent]=this.heap[index]; this.heap[index]=tmp; this.\_percolateUp(parent);

}

};

\_maxHeapify(index)

{ let left=(index\*2)+1; let right=(index\*2)+2; let largest=index; if ((this.elements > left) && (this.heap[largest] < this.heap[left])) { largest = left;

} if ((this.elements > right) && (this.heap[largest] < this.heap[right])) largest = right; else if(largest!==index)

{ const tmp = this.heap[largest]; this.heap[largest] = this.heap[index]; this.heap[index] = tmp; this.\_maxHeapify(largest);

}

}; buildHeap(arr){ this.heap = arr; this.elements = this.heap.length; for (let i = this.heap.length - 1; i >= 0; i--) { this.\_maxHeapify(i);

}

};

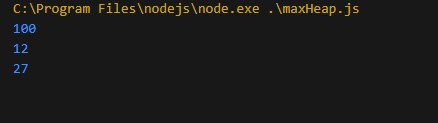
};

let heap = new maxHeap(); heap.insert(12); heap.insert(10); heap.insert(-10); heap.insert(100); heap.insert(-15);

console.log(heap.getMax()); let newheap = new maxHeap();

arr = [12, 6, 8, 3, 16, 4, 27]; newheap.buildHeap(arr); console.log(newheap.getMax()); newheap.removeMax(); console.log(newheap.getMax());

**output :**



**Q12. Write a JS program for implementation of Hashing.**

class HashTable

{ constructor(size=50){ this.buckets=new Array(size); this.size=size;

} hash(key){ return key.toString().length % this.size;

} setItem(key,value){ let index=this.hash(key); if(!this.buckets[index]){ this.buckets[index]=[];

} this.buckets[index].push([key,value]); return index;

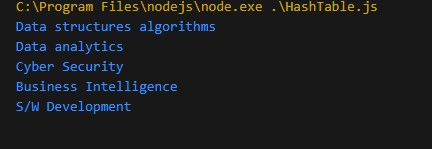
} getItem(key){ let index=this.hash(key); if(!this.buckets[index]) return null; for(let bucket of this.buckets[index]){ if(bucket[0]===key){ return bucket[1];

}

}

} } const hashTable=new HashTable(); hashTable.setItem("bk101","Data structures algorithms"); hashTable.setItem("bk108","Data analytics"); hashTable.setItem("bk200","Cyber Security"); hashTable.setItem("bk259","Business Intelligence"); hashTable.setItem("bk330","S/W Development"); hashTable.getItem("bk101"); console.log(hashTable.getItem("bk101")); hashTable.getItem("bk108"); console.log(hashTable.getItem("bk108")); hashTable.getItem("bk200"); console.log(hashTable.getItem("bk200")); hashTable.getItem("bk259"); console.log(hashTable.getItem("bk259")); hashTable.getItem("bk330"); console.log(hashTable.getItem("bk330"));

**output :**



**Q13. . Write a JS program Rain water Trapping (Practical based on**

**Brute Force technique)**

function maxWater(arr,n)

{

let res=0; for(let i=1;i<n-1;i++)

{ let left=arr[i]; for(let j=0;j<i;j++)

{ left=Math.max(left,arr[j]);

} let right=arr[i]; for(let j=i+1;j<n;j++)

{

right=Math.max(right,arr[j]);

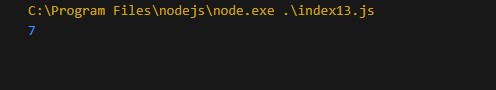
} res+=Math.min(left,right)-arr[i];

} return res;

}

let arr=[2,0,3,0,1,0,2]; let n=arr.length; console.log(maxWater(arr,n));

**output :**



**Q14. Write a JS program Jump Game.( Practical based on Greedy Algorithm).**

function minJumps(n,Jumps)

{ var i,j var dp=[n] dp[0]=0

for(i=1;i<n;i++)

dp[i]=Number.MAX\_VALUE

for(i=0;i<n-1;i++)

{

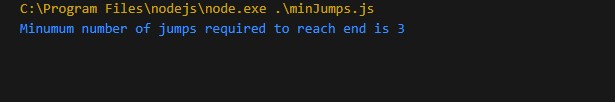
for(j=1;j<=jumps[i] && i+j<n;j++) { dp[i+j]=Math.min(dp[i+j],1+dp[i]);

} }

return dp[n-1]

} var jumps=[2,3,1,1,4,5,4] var n=jumps.length console.log("Minumum number of jumps required to reach end is",minJumps(7,jumps))

**output:**



**Q15. Write JS program for Binary Search(practical based on Divide and Conquer technique)**

function binarySearch(arr,l,r,x)

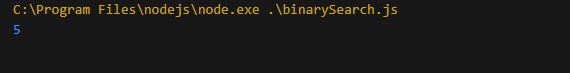
{

if(r>=1) { var mid=Math.floor((l+r)/2); //element is present at mid if(arr[mid]==x) return mid; //smaller than mid if(arr[mid]>x) return binarySearch(arr,l,mid-1,x) else return binarySearch(arr,mid+1,r,x)

} return -1;

}

let arr=[1,3,5,7,8,9] let x=9 console.log(binarySearch(arr,0,arr.length-1,x))



**Q16. Write JS program for finding out power set(practical based on Backtracking)**

function subset(arr)

{

result=[] if(arr==null || arr.length==0) return result subset=[] backtrack(arr,subset,0)

}

function backtrack(arr,subset,start)

{

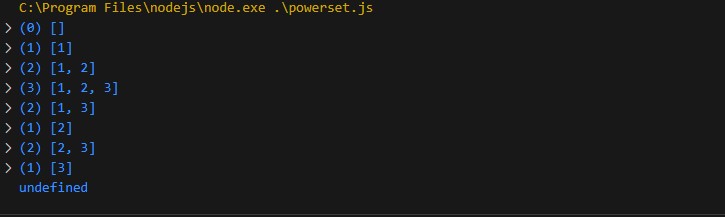
console.log(subset) for(var i=start;i<arr.length;i++)

{ subset.push(arr[i]) backtrack(arr,subset,i+1) subset.pop(subset.length-1)

}

}

console.log(subset([1,2,3]))



**Q17. Write JS program for BST.**

function binarySearch(arr,l,r,x)

{

if(r>=1) { var mid=Math.floor((l+r)/2); //element is present at mid if(arr[mid]==x) return mid; //smaller than mid if(arr[mid]>x) return binarySearch(arr,l,mid-1,x) else return binarySearch(arr,mid+1,r,x)

} return -1;

}

let arr=[1,3,5,7,8,9] let x=9 console.log(binarySearch(arr,0,arr.length-1,x))



**Q18. Write JS program for compute a^n where n is positive integer using fast power method**

function modular(base,exp,mod)

{

var res=1

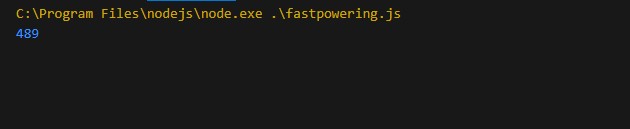
while(exp>0)

{ if(exp%2==1) res=(res\*base)%mod exp=exp>>1 base=(base \* base)%mod

} return res

}

console.log(modular(3,218,1000))



**Q19. Write JS program for finding GCD using Euclidean algorithm**

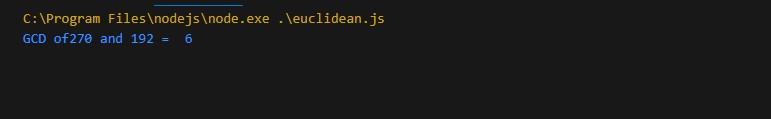
function gcd(a,b)

{

if(a==0) return b return gcd(b%a,a)

}

var a=270,b=192 console.log("GCD of"+a+" and "+b+" = ",gcd(a,b))



**Q20. Write JS program to create a Pascal’s Triangle.**

function PascalTriangle(numRows)

{

if(numRows==0) return [] if(numRows==1)return [[1]] let result=[]

for(let row=1;row<=numRows;row++)

{ let arr=[] for(let col=0;col<row;col++)

{ if(col===0 || col===row-1)

{ arr.push(1)

} else {

arr.push((result[row-2][col-1]+result[row-2][col]))

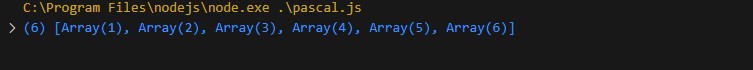
} } result.push(arr)

}

return result

}

console.log(PascalTriangle(6))



**Q21. Write JS program Dijkstra shortest path algorithm using Prim’s Algorithm.**

**Dijkstra algorithm :**

function minimumDist(dist,Tset){

var min=Number.MAX\_VALUE,index

for(var i=0;i<n;i++){ if(Tset[i]==false && dist[i]<=min){ min=dist[i] index=i

} } return index

}

function Dijkstra(graph,src,n){ var dist=[n];

var Tset=[n]; for(var i=0;i<4;i++){

dist[i]=Number.MAX\_VALUE

Tset[i]=false

} dist[src]=0 for(var i=0;i<n;i++){

var m=minimumDist(dist,Tset); Tset[m]=true for(var i=0;i<n;i++){

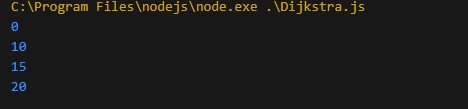
if(!Tset[i] && graph[m][i] && dist[m]!=Number.MAX\_VALUE && dist[m]+graph[m][i]<dist[i]) dist[i]=dist[m]+graph[m][i]

} } return dist

}

var graph=[[0,10,15,20],[10,0,35,25],[15,35,0,30],[20,25,30,0]] var s=0 var n=4 var dist=Dijkstra(graph,s,n) for(var i=0;i<n;i++) console.log(dist[i])

**output:**



**Prim’s algorithm :**

class Graph { constructor(vertices) { this.vertices = vertices; this.adjacencyList = new Map();

} addEdge(vertex1, vertex2, weight) { if (!this.adjacencyList.has(vertex1)) { this.adjacencyList.set(vertex1, []);

}

if (!this.adjacencyList.has(vertex2)) { this.adjacencyList.set(vertex2, []);

} this.adjacencyList.get(vertex1).push({ node: vertex2, weight }); this.adjacencyList.get(vertex2).push({ node: vertex1, weight });

} primMST(startVertex) { const visited = new Set(); const result = [];

const priorityQueue = new PriorityQueue();

// Add starting vertex to the priority queue with priority 0 priorityQueue.enqueue(startVertex, 0);

while (!priorityQueue.isEmpty()) { const currentVertex = priorityQueue.dequeue().data;

if (!visited.has(currentVertex)) { visited.add(currentVertex);

// Add the edges of the current vertex to the priority queue const edges = this.adjacencyList.get(currentVertex); for (const edge of edges) { const { node, weight } = edge; if (!visited.has(node)) { priorityQueue.enqueue(node, weight);

}

}

// Add the current edge to the result if (result.length < this.vertices - 1) {

const minEdge = { start: currentVertex, end: priorityQueue.peek().data, weight: priorityQueue.peek().priority,

}; result.push(minEdge);

}

}

} return result;

}

} class PriorityQueue { constructor() { this.queue = [];

} enqueue(data, priority) { this.queue.push({ data, priority }); this.sort();

} dequeue() { return this.queue.shift();

} peek() { return this.queue[0]; } isEmpty() { return this.queue.length === 0;

} sort() { this.queue.sort((a, b) => a.priority - b.priority);

}

}

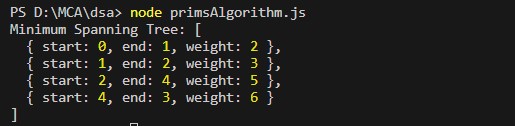
// Example usage

// Example usage with sample input and output const graph = new Graph(5);

graph.addEdge(0, 1, 2); graph.addEdge(0, 3, 6); graph.addEdge(1, 2, 3); graph.addEdge(1, 3, 8); graph.addEdge(1, 4, 5); graph.addEdge(2, 4, 7); graph.addEdge(3, 4, 9);

const minimumSpanningTree = graph.primMST(0); console.log("Minimum Spanning Tree:", minimumSpanningTree);

**output:**



**Kruskal’s Algorithm -**

class Graph { constructor(vertices) { this.vertices = vertices; this.edges = [];

} addEdge(src, dest, weight) { this.edges.push({ src, dest, weight });

}

kruskalMST() { this.edges.sort((a, b) => a.weight - b.weight);

const result = []; const parent = Array.from({ length: this.vertices }, (\_, index) => index);

const find = (vertex) => { if (parent[vertex] !== vertex) { parent[vertex] = find(parent[vertex]);

} return parent[vertex];

};

const union = (x, y) => { parent[find(x)] = find(y);

}; this.edges.forEach(({ src, dest, weight }) => { if (find(src) !== find(dest)) { result.push({ src, dest, weight }); union(src, dest);

}

}); return result;

}

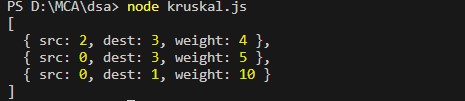
}

// Example usage const graph = new Graph(4);

graph.addEdge(0, 1, 10); graph.addEdge(0, 2, 6); graph.addEdge(0, 3, 5); graph.addEdge(1, 3, 15); graph.addEdge(2, 3, 4);

const minimumSpanningTree = graph.kruskalMST(); console.log(minimumSpanningTree);

**output:**



**Q22. Write JS program for sorting array using quick sort.**

function quickSort(arr) { if (arr.length <= 1) { return arr;

} const pivot = arr[0]; const left = []; const right = [];

for (let i = 1; i < arr.length; i++) { if (arr[i] < pivot) { left.push(arr[i]); } else { right.push(arr[i]);

}

}

return [...quickSort(left), pivot, ...quickSort(right)];

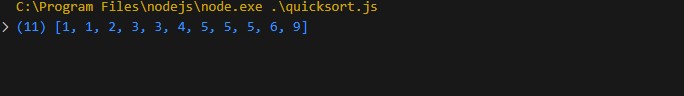
}

// Example usage:

const unsortedArray = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]; const sortedArray = quickSort(unsortedArray);

console.log(sortedArray);

**output:**



**Q23. Write JS program for staircase problem.**

function fibo(n)

{

if(n<=1) return n else return fibo(n-2)+fibo(n-1)

}

function count\_ways(s)

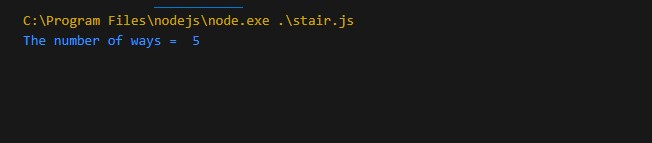
{

return fibo(s+1)

}

console.log("The number of ways = ",count\_ways(4))

**output:**



**Q24. Write JS program for tower of hanoi.**

function towerOfHanoi(n,A,B,C)

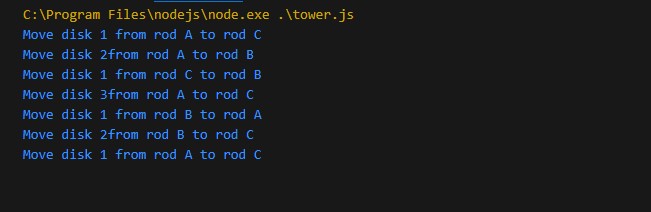
{

if(n==1) { console.log("Move disk 1 from rod "+A+" to rod "+B) return } towerOfHanoi(n-1,A,C,B) console.log("Move disk "+n+"from rod "+A+" to rod "+B) towerOfHanoi(n-1,C,B,A)

}

var n=3 towerOfHanoi(n,'A','C','B')

**output:**



**Q25. Write JS program for powerset.**

function subset(arr)

{

result=[] if(arr==null || arr.length==0) return result subset=[] backtrack(arr,subset,0)

}

function backtrack(arr,subset,start)

{

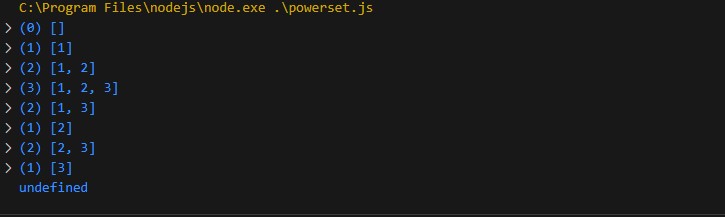
console.log(subset) for(var i=start;i<arr.length;i++)

{ subset.push(arr[i]) backtrack(arr,subset,i+1) subset.pop(subset.length-1) }

}

console.log(subset([1,2,3]))

**output:**



**Q26. Write JS program for binarysearch.**

function binarySearch(arr,l,r,x)

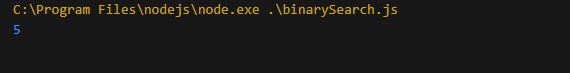
{

if(r>=1) { var mid=Math.floor((l+r)/2); //element is present at mid if(arr[mid]==x) return mid; //smaller than mid if(arr[mid]>x) return binarySearch(arr,l,mid-1,x) else return binarySearch(arr,mid+1,r,x) } return -1;

}

let arr=[1,3,5,7,8,9] let x=9 console.log(binarySearch(arr,0,arr.length-1,x))

**output:**



**Q27. Write JS program for Euclidian algorithm.**

function gcd(a,b)

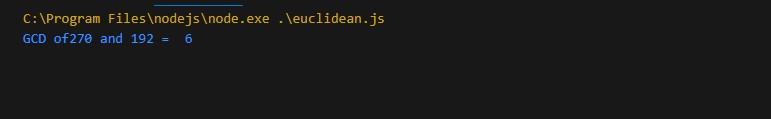
{

if(a==0) return b return gcd(b%a,a)

}

var a=270,b=192 console.log("GCD of"+a+" and "+b+" = ",gcd(a,b))

**output:**



**Q28. Write JS program for fastpowering algorithm.**

function modular(base,exp,mod)

{

var res=1 while(exp>0)

{ if(exp%2==1) res=(res\*base)%mod exp=exp>>1 base=(base \* base)%mod

} return res

}

console.log(modular(3,218,1000))

**output:**

