**PRACTICAL NO:1**

**Title: Implement depth first search algorithm and Breadth First Search algorithm, Use an undirected graph and develop a recursive algorithm for searching all the vertices of a graph or tree data structure.**

**Code:**

#include<stdio.h>

#include<stdlib.h>

int visited[7] = {0,0,0,0,0,0,0};

int A[7][7] = {

{0,1,1,1,0,0,0},

{1,0,1,0,0,0,0},

{1,1,0,1,1,0,0},

{1,0,1,0,1,0,0},

{0,0,1,1,0,1,1},

{0,0,0,0,1,0,0},

{0,0,0,0,1,0,0}

};

void DFS(int i) {

printf("%d ", i);

visited[i] = 1;

for (int j = 0; j < 7; j++) {

if (A[i][j] == 1 && !visited[j]) {

DFS(j);

}

}

}

int main() {

// DFS Implementation

DFS(0);

return 0;

}

**PRACTICAL NO:2**

**Title: Implement A star Algorithm for any game search problem.**

**Code:**

class Main {

static class AStarNode {

// Node attributes

int s\_x; // x-coordinate

int s\_y; // y-coordinate

int s\_g; // cost from start node to this node

int s\_h; // heuristic estimate from this node to end node

int s\_style; // Node type: start, end, obstacle

AStarNode s\_parent; // Parent node

boolean s\_is\_in\_closetable; // Flag indicating whether node is in close table

boolean s\_is\_in\_opentable; // Flag indicating whether node is in open table

}

static AStarNode[][] map\_maze = new AStarNode[10][10]; // Node array

static AStarNode[] open\_table = new AStarNode[100]; // Open table

static AStarNode[] close\_table = new AStarNode[100]; // Close table

static AStarNode[] path\_stack = new AStarNode[100]; // Stack to save the path

static int open\_node\_count = 0; // Number of nodes in the open table

static int close\_node\_count = 0; // Number of nodes in the close table

static int top = -1; // Top of stack

// Swap two elements in the open table

static void swap(int idx1, int idx2) {

AStarNode tmp = open\_table[idx1];

open\_table[idx1] = open\_table[idx2];

open\_table[idx2] = tmp;

}

// Heap adjustment

static void adjust\_heap(int nIndex) {

int curr = nIndex;

int child = curr \* 2 + 1;

int parent = (curr - 1) / 2;

if (nIndex < 0 || nIndex >= open\_node\_count) {

return;

}

while (child < open\_node\_count) {

if (child + 1 < open\_node\_count && (open\_table[child].s\_g + open\_table[child].s\_h) > (open\_table[child + 1].s\_g + open\_table[child + 1].s\_h)) {

++child;

}

if ((open\_table[curr].s\_g + open\_table[curr].s\_h) <= (open\_table[child].s\_g + open\_table[child].s\_h)) {

break;

} else {

swap(child, curr);

curr = child;

child = curr \* 2 + 1;

}

}

while (curr != 0) {

if ((open\_table[curr].s\_g + open\_table[curr].s\_h) >= (open\_table[parent].s\_g + open\_table[parent].s\_h)) {

break;

} else {

swap(curr, parent);

curr = parent;

parent = (curr - 1) / 2;

}

}

}

// Insert node to open table

static void insert\_to\_opentable(int x, int y, AStarNode curr\_node, AStarNode end\_node, int w) {

if (map\_maze[x][y].s\_style != 3) { // Not an obstacle

if (!map\_maze[x][y].s\_is\_in\_closetable) { // Not in the close table

if (map\_maze[x][y].s\_is\_in\_opentable) { // In the open table

if (map\_maze[x][y].s\_g > curr\_node.s\_g + w) { // More optimized path

map\_maze[x][y].s\_g = curr\_node.s\_g + w;

map\_maze[x][y].s\_parent = curr\_node;

int i;

for (i = 0; i < open\_node\_count; ++i) {

if (open\_table[i].s\_x == map\_maze[x][y].s\_x && open\_table[i].s\_y == map\_maze[x][y].s\_y) {

break;

}

}

adjust\_heap(i); // Adjust heap

}

} else { // Not in open

map\_maze[x][y].s\_g = curr\_node.s\_g + w;

map\_maze[x][y].s\_h = Math.abs(end\_node.s\_x - x) + Math.abs(end\_node.s\_y - y);

map\_maze[x][y].s\_parent = curr\_node;

map\_maze[x][y].s\_is\_in\_opentable = true;

open\_table[open\_node\_count++] = map\_maze[x][y];

}

}

}

}

// Get neighbors of a node

static void get\_neighbors(AStarNode curr\_node, AStarNode end\_node) {

int x = curr\_node.s\_x;

int y = curr\_node.s\_y;

if ((x + 1) >= 0 && (x + 1) < 10 && y >= 0 && y < 10) {

insert\_to\_opentable(x + 1, y, curr\_node, end\_node, 10);

}

if ((x - 1) >= 0 && (x - 1) < 10 && y >= 0 && y < 10) {

insert\_to\_opentable(x - 1, y, curr\_node, end\_node, 10);

}

if (x >= 0 && x < 10 && (y + 1) >= 0 && (y + 1) < 10) {

insert\_to\_opentable(x, y + 1, curr\_node, end\_node, 10);

}

if (x >= 0 && x < 10 && (y - 1) >= 0 && (y - 1) < 10) {

insert\_to\_opentable(x, y - 1, curr\_node, end\_node, 10);

}

if ((x + 1) >= 0 && (x + 1) < 10 && (y + 1) >= 0 && (y + 1) < 10) {

insert\_to\_opentable(x + 1, y + 1, curr\_node, end\_node, 14);

}

if ((x + 1) >= 0 && (x + 1) < 10 && (y - 1) >= 0 && (y - 1) < 10) {

insert\_to\_opentable(x + 1, y - 1, curr\_node, end\_node, 14);

}

if ((x - 1) >= 0 && (x - 1) < 10 && (y + 1) >= 0 && (y + 1) < 10) {

insert\_to\_opentable(x - 1, y + 1, curr\_node, end\_node, 14);

}

if ((x - 1) >= 0 && (x - 1) < 10 && (y - 1) >= 0 && (y - 1) < 10) {

insert\_to\_opentable(x - 1, y - 1, curr\_node, end\_node, 14);

}

}

public static void main(String[] args) {

// Define the map array, start\_node, and end\_node

AStarNode start\_node = null;

AStarNode end\_node = null;

AStarNode curr\_node;

int is\_found;

int[][] maze = { // Define your maze here

{1, 0, 0, 3, 0, 3, 0, 0, 0, 0},

{0, 0, 3, 0, 0, 3, 0, 3, 0, 3},

{3, 0, 0, 0, 0, 3, 3, 3, 0, 3},

{3, 0, 3, 0, 0, 0, 0, 0, 0, 3},

{3, 0, 0, 0, 0, 3, 0, 0, 0, 3},

{3, 0, 0, 3, 0, 0, 0, 3, 0, 3},

{3, 0, 0, 0, 0, 3, 3, 0, 0, 0},

{0, 0, 2, 0, 0, 0, 0, 0, 0, 0},

{3, 3, 3, 0, 0, 3, 0, 3, 0, 3},

{3, 0, 0, 0, 0, 3, 3, 3, 0, 3}

};

// Initialize map\_maze array

for (int i = 0; i < 10; ++i) {

for (int j = 0; j < 10; ++j) {

map\_maze[i][j] = new AStarNode();

map\_maze[i][j].s\_g = 0;

map\_maze[i][j].s\_h = 0;

map\_maze[i][j].s\_is\_in\_closetable = false;

map\_maze[i][j].s\_is\_in\_opentable = false;

map\_maze[i][j].s\_style = maze[i][j];

map\_maze[i][j].s\_x = i;

map\_maze[i][j].s\_y = j;

if (map\_maze[i][j].s\_style == 1) {

start\_node = map\_maze[i][j];

} else if (map\_maze[i][j].s\_style == 2) {

end\_node = map\_maze[i][j];

}

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

}

System.out.println();

}

open\_table[open\_node\_count++] = start\_node;

start\_node.s\_is\_in\_opentable = true;

start\_node.s\_g = 0;

start\_node.s\_h = Math.abs(end\_node.s\_x - start\_node.s\_x) + Math.abs(end\_node.s\_y - start\_node.s\_y);

start\_node.s\_parent = null;

if (start\_node.s\_x == end\_node.s\_x && start\_node.s\_y == end\_node.s\_y) {

System.out.println("Start point == End point!");

return;

}

is\_found = 0;

while (true) {

curr\_node = open\_table[0];

open\_table[0] = open\_table[--open\_node\_count];

adjust\_heap(0);

close\_table[close\_node\_count++] = curr\_node;

curr\_node.s\_is\_in\_closetable = true;

if (curr\_node.s\_x == end\_node.s\_x && curr\_node.s\_y == end\_node.s\_y) {

is\_found = 1;

break;

}

get\_neighbors(curr\_node, end\_node);

if (open\_node\_count == 0) {

is\_found = 0;

break;

}

}

if (is\_found == 1) {

curr\_node = end\_node;

while (curr\_node != null) {

path\_stack[++top] = curr\_node;

curr\_node = curr\_node.s\_parent;

}

System.out.println("Path found:");

while (top >= 0) {

if (top > 0) {

System.out.print("(" + path\_stack[top].s\_x + "," + path\_stack[top].s\_y + ")->");

} else {

System.out.print("(" + path\_stack[top].s\_x + "," + path\_stack[top].s\_y + ")");

}

top--;

}

} else {

System.out.println("Path not found");

}

System.out.println();

}

}

**PRACTICAL NO:3**

**Title: Implement Greedy search algorithm for any of the following application:**

**I. Selection Sort**

**II. Minimum Spanning Tree**

**III. Single-Source Shortest Path Problem**

**IV. Job Scheduling Problem V.Prim's Minimal Spanning Tree Algorithm**

**VI. Kruskal's Minimal Spanning Tree Algorithm**

**VII. Dijkstra's Minimal Spanning Tree Algorithm**

**Code:**

import java.io.\*;

import java.util.\*;

class DisjointSet {

List<Integer> rank = new ArrayList<>();

List<Integer> parent = new ArrayList<>();

List<Integer> size = new ArrayList<>();

public DisjointSet(int n) {

for (int i = 0; i <= n; i++) {

rank.add(0);

parent.add(i);

size.add(1);

}

}

public int findUPar(int node) {

if (node == parent.get(node)) {

return node;

}

int ulp = findUPar(parent.get(node));

parent.set(node, ulp);

return parent.get(node);

}

public void unionByRank(int u, int v) {

int ulp\_u = findUPar(u);

int ulp\_v = findUPar(v);

if (ulp\_u == ulp\_v) return;

if (rank.get(ulp\_u) < rank.get(ulp\_v)) {

parent.set(ulp\_u, ulp\_v);

} else if (rank.get(ulp\_v) < rank.get(ulp\_u)) {

parent.set(ulp\_v, ulp\_u);

} else {

parent.set(ulp\_v, ulp\_u);

int rankU = rank.get(ulp\_u);

rank.set(ulp\_u, rankU + 1);

}

}

public void unionBySize(int u, int v) {

int ulp\_u = findUPar(u);

int ulp\_v = findUPar(v);

if (ulp\_u == ulp\_v) return;

if (size.get(ulp\_u) < size.get(ulp\_v)) {

parent.set(ulp\_u, ulp\_v);

size.set(ulp\_v, size.get(ulp\_v) + size.get(ulp\_u));

} else {

parent.set(ulp\_v, ulp\_u);

size.set(ulp\_u, size.get(ulp\_u) + size.get(ulp\_v));

}

}

}

class Edge implements Comparable<Edge> {

int src, dest, weight;

Edge(int \_src, int \_dest, int \_wt) {

this.src = \_src; this.dest = \_dest; this.weight = \_wt;

}

// Comparator function used for

// sorting edgesbased on their weight

public int compareTo(Edge compareEdge) {

return this.weight - compareEdge.weight;

}

};

class Solution {

//Function to find sum of weights of edges of the Minimum Spanning Tree.

static int spanningTree(int V,

ArrayList<ArrayList<ArrayList<Integer>>> adj) {

List<Edge> edges = new ArrayList<Edge>();

// O(N + E)

for (int i = 0; i < V; i++) {

for (int j = 0; j < adj.get(i).size(); j++) {

int adjNode = adj.get(i).get(j).get(0);

int wt = adj.get(i).get(j).get(1);

int node = i;

Edge temp = new Edge(i, adjNode, wt);

edges.add(temp);

}

}

DisjointSet ds = new DisjointSet(V);

// M log M

Collections.sort(edges);

int mstWt = 0;

// M x 4 x alpha x 2

for (int i = 0; i < edges.size(); i++) {

int wt = edges.get(i).weight;

int u = edges.get(i).src;

int v = edges.get(i).dest;

if (ds.findUPar(u) != ds.findUPar(v)) {

mstWt += wt;

ds.unionBySize(u, v);

}

}

return mstWt;

}

}

class Main {

public static void main (String[] args) {

int V = 5;

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

int[][] edges = {{0, 1, 2}, {0, 2, 1}, {1, 2, 1}, {2, 3, 2}, {3, 4, 1}, {4, 2, 2}};

for (int i = 0; i < V; i++) {

adj.add(new ArrayList<ArrayList<Integer>>());

}

for (int i = 0; i < 6; i++) {

int u = edges[i][0];

int v = edges[i][1];

int w = edges[i][2];

ArrayList<Integer> tmp1 = new ArrayList<Integer>();

ArrayList<Integer> tmp2 = new ArrayList<Integer>();

tmp1.add(v);

tmp1.add(w);

tmp2.add(u);

tmp2.add(w);

adj.get(u).add(tmp1);

adj.get(v).add(tmp2);

}

Solution obj = new Solution();

int mstWt = obj.spanningTree(V, adj);

System.out.println("The sum of all the edge weights: " + mstWt);

}

}

**PRACTICAL NO: 4**

**TITLE: Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph colouring problem**

**Code:**

import java.util.\*;

class Main {

public static List < List < String >> solveNQueens(int n) {

char[][] board = new char[n][n];

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

board[i][j] = '.';

List < List < String >> res = new ArrayList < List < String >> ();

dfs(0, board, res);

return res;

}

static boolean validate(char[][] board, int row, int col) {

int duprow = row;

int dupcol = col;

while (row >= 0 && col >= 0) {

if (board[row][col] == 'Q') return false;

row--;

col--;

}

row = duprow;

col = dupcol;

while (col >= 0) {

if (board[row][col] == 'Q') return false;

col--;

}

row = duprow;

col = dupcol;

while (col >= 0 && row < board.length) {

if (board[row][col] == 'Q') return false;

col--;

row++;

}

return true;

}

static void dfs(int col, char[][] board, List < List < String >> res) {

if (col == board.length) {

res.add(construct(board));

return;

}

for (int row = 0; row < board.length; row++) {

if (validate(board, row, col)) {

board[row][col] = 'Q';

dfs(col + 1, board, res);

board[row][col] = '.';

}

}

}

static List < String > construct(char[][] board) {

List < String > res = new LinkedList < String > ();

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

String s = new String(board[i]);

res.add(s);

}

return res;

}

public static void main(String args[]) {

int N = 8;

List < List < String >> queen = solveNQueens(N);

int i = 1;

for (List < String > it: queen) {

System.out.println("Arrangement " + i);

for (String s: it) {

System.out.println(s);

}

System.out.println();

i += 1;

}

}

}

**PRACTICAL NO:5**

**Title: Develop an elementary chatbot for any suitable customer interaction**

import java.util.\*;

public class Main {

public static void main(String[] args) {

Map<String, String> map = new HashMap<>();

map.put("hi", "Hello, What can Help you");

map.put("hello", "Hello, What can Help you");

map.put("what is your college address", "Wagholi, Pune");

Scanner sc = new Scanner(System.in);

String choice;

do {

int ch;

System.out.println("1) Ask Question");

System.out.println("2) Exit");

ch = sc.nextInt();

sc.nextLine(); // Consume newline

String que;

switch (ch) {

case 1:

System.out.println("Enter your Question : ");

que = sc.nextLine(); // Read entire line

que = que.toLowerCase();

System.out.println("Ans : ");

if (map.containsKey(que)) {

System.out.println(map.get(que));

} else {

System.out.println("I can't answer this question");

}

break;

case 2:

System.exit(0);

break;

}

System.out.println("Do you Want to continue : (Y|N)");

choice = sc.next();

} while (choice.equalsIgnoreCase("Y"));

sc.close();

}

}

**PRACTICAL NO 7**

**Title: Write a Java/C/C++/Python program that contains a string (char pointer) with a value \Hello World’. The program should AND or and XOR each character in this string with 127 and display the result.**

**Code:**

public class Main {

public static void main(String[] args) {

String str = "Hello World"; // String containing \Hello World

System.out.println("Original String: " + str);

// Perform AND and XOR operations on each character with 127

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

char ch = str.charAt(i);

// AND operation with 127

char andResult = (char) (ch & 127);

System.out.println("Character '" + ch + "' AND 127: " + (int) andResult);

// XOR operation with 127

char xorResult = (char) (ch ^ 127);

System.out.println("Character '" + ch + "' XOR 127: " + (int) xorResult);

}

}

}

**PRACTICAL NO:8**

**Title: Write a Java/C/C++/Python program to perform encryption and decryption using the method of Transposition technique**

**Code:**

import java.util.Scanner;

public class Main {

public static String encryptMessage(String msg, String key) {

StringBuilder cipher = new StringBuilder();

int k\_indx = 0;

int msg\_len = msg.length();

char[] msg\_lst = msg.toCharArray();

char[] key\_lst = key.toUpperCase().toCharArray();

int col = key.length();

int row = (int) Math.ceil((double) msg\_len / col);

int fill\_null = (row \* col) - msg\_len;

for (int i = 0; i < fill\_null; i++) {

msg += '\_';

}

char[][] matrix = new char[row][col];

for (int i = 0; i < msg\_len; i++) {

matrix[i / col][i % col] = msg\_lst[i];

}

for (int i = 0; i < col; i++) {

int curr\_idx = key.indexOf(key\_lst[k\_indx]);

for (int j = 0; j < row; j++) {

cipher.append(matrix[j][curr\_idx]);

}

k\_indx++;

}

return cipher.toString();

}

public static String decryptMessage(String cipher, String key) {

StringBuilder msg = new StringBuilder();

int k\_indx = 0;

int msg\_indx = 0;

int msg\_len = cipher.length();

char[] msg\_lst = cipher.toCharArray();

int col = key.length();

int row = (int) Math.ceil((double) msg\_len / col);

char[] key\_lst = key.toUpperCase().toCharArray();

char[][] dec\_cipher = new char[row][col];

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

dec\_cipher[i][j] = ' ';

}

}

for (int i = 0; i < col; i++) {

int curr\_idx = key.indexOf(key\_lst[k\_indx]);

for (int j = 0; j < row; j++) {

dec\_cipher[j][curr\_idx] = msg\_lst[msg\_indx++];

}

k\_indx++;

}

for (int i = 0; i < row; i++) {

for (int j = 0; j < col; j++) {

if (dec\_cipher[i][j] != '\_') {

msg.append(dec\_cipher[i][j]);

}

}

}

return msg.toString();

}

public static String checkKey(String key) {

// Check for duplicate characters in the key

if (key.chars().distinct().count() != key.length()) {

throw new IllegalArgumentException("Key should not contain duplicate characters.");

}

return key;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the key: ");

String key = scanner.nextLine().toUpperCase();

try {

key = checkKey(key);

} catch (IllegalArgumentException e) {

System.out.println("ERROR! " + e.getMessage());

return;

}

System.out.print("Enter the message: ");

String msg = scanner.nextLine().toUpperCase();

String cipher = encryptMessage(msg, key);

System.out.println("Encrypted Message: " + cipher);

String decryptedMsg = decryptMessage(cipher, key);

System.out.println("Decrypted Message: " + decryptedMsg);

scanner.close();

}

}

**PRACTICAL NO 9**

**Title:**

**Write a Java/C/C++/Python program to implement DES algorithm. Theory: DES Algorithm: The DES (Data Encryption Standard) algorithm is a symmetric-key block cipher created in the early 1970s by an IBM team and adopted by the National Institute of Standards and Technology (NIST). The algorithm takes the plain text in 64-bit blocks and converts them into ciphertext using 48-bit keys.Since it’s a symmetric-key algorithm, it employs the same key in both encrypting and decrypting the data. If it were an asymmetrical algorithm, it would use different keys for encryption and decryption**

**Code 1:**

import javax.crypto.\*;

import javax.crypto.spec.\*;

import java.security.\*;

import java.util.Base64;

public class Main {

public static void main(String[] args) {

String password = "Password";

byte[] salt = {(byte) 0x28, (byte) 0xAB, (byte) 0xBC, (byte) 0xCD, (byte) 0xDE, (byte) 0xEF, (byte) 0x00, (byte) 0x33};

// Concatenate password and salt

byte[] key = new byte[password.length() + salt.length];

System.arraycopy(password.getBytes(), 0, key, 0, password.length());

System.arraycopy(salt, 0, key, password.length(), salt.length);

// Calculate MD5 hash of the key

byte[] hash = null;

try {

MessageDigest md = MessageDigest.getInstance("MD5");

hash = md.digest(key);

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

}

if (hash != null) {

// Split the hash into DES key and IV

byte[] dk = new byte[8];

byte[] iv = new byte[8];

System.arraycopy(hash, 0, dk, 0, 8);

System.arraycopy(hash, 8, iv, 0, 8);

// Create DES cipher instance

try {

Cipher crypter = Cipher.getInstance("DES/CBC/PKCS5Padding");

SecretKeySpec keySpec = new SecretKeySpec(dk, "DES");

IvParameterSpec ivSpec = new IvParameterSpec(iv);

crypter.init(Cipher.ENCRYPT\_MODE, keySpec, ivSpec);

String plainText = "I see you";

System.out.println("The plain text is : " + plainText);

// Add padding if needed

int paddingLength = 8 - (plainText.length() % 8);

if (paddingLength != 8) {

plainText += String.valueOf((char) 0).repeat(paddingLength);

}

// Encrypt the plaintext

byte[] ciphertext = crypter.doFinal(plainText.getBytes());

// Encode the ciphertext using Base64

String encodedString = Base64.getEncoder().encodeToString(ciphertext);

System.out.println("The encoded string is : " + encodedString);

} catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException | IllegalBlockSizeException | BadPaddingException | InvalidAlgorithmParameterException e) {

e.printStackTrace();

}

}

}

}

**Code 2:**

import javax.crypto.\*;

import javax.crypto.spec.\*;

import java.security.\*;

import java.util.Base64;

public class Main {

public static void main(String[] args) {

String password = "Password";

byte[] salt = {(byte) 0x28, (byte) 0xAB, (byte) 0xBC, (byte) 0xCD, (byte) 0xDE, (byte) 0xEF, (byte) 0x00, (byte) 0x33};

// Concatenate password and salt

byte[] key = new byte[password.length() + salt.length];

System.arraycopy(password.getBytes(), 0, key, 0, password.length());

System.arraycopy(salt, 0, key, password.length(), salt.length);

// Calculate MD5 hash of the key

byte[] hash = null;

try {

MessageDigest md = MessageDigest.getInstance("MD5");

hash = md.digest(key);

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

}

if (hash != null) {

// Split the hash into DES key and IV

byte[] dk = new byte[8];

byte[] iv = new byte[8];

System.arraycopy(hash, 0, dk, 0, 8);

System.arraycopy(hash, 8, iv, 0, 8);

// Create DES cipher instance

try {

Cipher crypter = Cipher.getInstance("DES/CBC/PKCS5Padding");

SecretKeySpec keySpec = new SecretKeySpec(dk, "DES");

IvParameterSpec ivSpec = new IvParameterSpec(iv);

crypter.init(Cipher.DECRYPT\_MODE, keySpec, ivSpec);

String encryptedString = "Fbim0Nyn6O/DZe8VP/UWrsoby8EYdymv";

System.out.println("The encrypted string is : " + encryptedString);

// Decode the Base32 encoded string

byte[] decodedBytes = Base64.getDecoder().decode(encryptedString);

// Decrypt the ciphertext

byte[] decryptedBytes = crypter.doFinal(decodedBytes);

// Convert decrypted bytes to string

String decryptedString = new String(decryptedBytes);

System.out.println("The decrypted string is : " + decryptedString);

} catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException | IllegalBlockSizeException | BadPaddingException | InvalidAlgorithmParameterException e) {

e.printStackTrace();

}

}

}

}

**PRACTICAL NO 10**

**Title: Write a Java/C/C++/Python program to implement AES algorithm. Theory: AES Algorithm: The AES algorithm (also known as the Rijndael algorithm) is a symmetrical block cipher algorithm that takes plain text in blocks of 128 bits and converts them to ciphertext using keys of 128, 192, and 256 bits. Since the AES algorithm is considered secure, it is in the worldwide standard**

**Code:**

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.Cipher;

import javax.crypto.spec.SecretKeySpec;

public class Main {

static String IV = "AAAAAAAAAAAAAAAA";

static String plaintext = "test text 123\u0000\u0000\u0000"; /\* Note null padding \*/

static String encryptionKey = "0123456789abcdef";

public static void main(String[] args) {

try {

System.out.println("== Java ==");

System.out.println("plain: " + plaintext);

byte[] cipher = encrypt(plaintext, encryptionKey);

System.out.print("cipher: ");

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

System.out.print(new Integer(cipher[i]) + " ");

System.out.println("");

String decrypted = decrypt(cipher, encryptionKey);

System.out.println("decrypt: " + decrypted);

} catch (Exception e) {

e.printStackTrace();

}

}

public static byte[] encrypt(String plainText, String encryptionKey) throws Exception {

Cipher cipher = Cipher.getInstance("AES/CBC/NoPadding", "SunJCE");

SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes("UTF-8"), "AES");

cipher.init(Cipher.ENCRYPT\_MODE, key, new IvParameterSpec(IV.getBytes("UTF-8")));

return cipher.doFinal(plainText.getBytes("UTF-8"));

}

public static String decrypt(byte[] cipherText, String encryptionKey) throws Exception {

Cipher cipher = Cipher.getInstance("AES/CBC/NoPadding", "SunJCE");

SecretKeySpec key = new SecretKeySpec(encryptionKey.getBytes("UTF-8"), "AES");

cipher.init(Cipher.DECRYPT\_MODE, key, new IvParameterSpec(IV.getBytes("UTF-8")));

return new String(cipher.doFinal(cipherText), "UTF-8");

}

}

**PRACTICAL NO 11**

**Title: Write a Java/C/C++/Python program to implement RSA algorithm. Theory: RSA Algorithm: RSA is a public-key cryptosystem that is widely used for secure data transmission. It is also one of theoldest. The acronym "RSA" comes from the surnames of Ron Rivest, Adi Shamir and Leonard Adleman, who publicly described the algorithm in 1977. RSA algorithm is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is givento everyone and Private key is kept private.**

**Code:**

import java.math.\*;

import java.util.\*;

class Main {

public static void main(String args[]) {

int p, q, n, z, d = 0, e, i;

// The number to be encrypted and decrypted

int msg = 12;

double c;

BigInteger msgback;

// 1st prime number p

p = 3;

// 2nd prime number q

q = 11;

n = p \* q;

z = (p - 1) \* (q - 1);

System.out.println("the value of z = " + z);

for (e = 2; e < z; e++) {

// e is for public key exponent

if (gcd(e, z) == 1) {

break;

}

}

System.out.println("the value of e = " + e);

for (i = 0; i <= 9; i++) {

int x = 1 + (i \* z);

// d is for private key exponent

if (x % e == 0) {

d = x / e;

break;

}

}

System.out.println("the value of d = " + d);

c = (Math.pow(msg, e)) % n;

System.out.println("Encrypted message is : " + c);

// converting int value of n to BigInteger

BigInteger N = BigInteger.valueOf(n);

// converting float value of c to BigInteger

BigInteger C = BigDecimal.valueOf(c).toBigInteger();

msgback = (C.pow(d)).mod(N);

System.out.println("Decrypted message is : " + msgback);

}

static int gcd(int e, int z) {

if (e == 0)

return z;

else

return gcd(z % e, e);

}

}

**PRACTICAL NO 12**

**Title: Calculate the message digest of a text using the MD5 algorithm in JAVA. Theory: MD5 Algorithm: Message Digest Algorithm 5 (MD5) is a cryptographic hash algorithm that can be used to create a 128-bit string value from an arbitrary length string. Although there has been insecurities identified with MD5, it is still widely used. MD5 is most commonly used to verify the integrity of files.**

**Code:**

import java.nio.charset.Charset;

import java.nio.charset.StandardCharsets;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class Main {

private static final Charset UTF\_8 = StandardCharsets.UTF\_8;

private static final String OUTPUT\_FORMAT = "%-20s:%s";

private static byte[] digest(byte[] input) {

MessageDigest md;

try {

md = MessageDigest.getInstance("MD5");

} catch (NoSuchAlgorithmException e) {

throw new IllegalArgumentException(e);

}

byte[] result = md.digest(input);

return result;

}

private static String bytesToHex(byte[] bytes) {

StringBuilder sb = new StringBuilder();

for (byte b : bytes) {

sb.append(String.format("%02x", b));

}

return sb.toString();

}

public static void main(String[] args) {

String pText = "Hello MD5";

System.out.println(String.format(OUTPUT\_FORMAT, "Input (string)", pText));

System.out.println(String.format(OUTPUT\_FORMAT, "Input (length)", pText.length()));

byte[] md5InBytes = digest(pText.getBytes(UTF\_8));

System.out.println(String.format(OUTPUT\_FORMAT, "MD5 (hex) ", bytesToHex(md5InBytes)));

// fixed length, 16 bytes, 128 bits.

System.out.println(String.format(OUTPUT\_FORMAT, "MD5 (length)", md5InBytes.length));

}

}