

JSPM's Bhivarabai Sawant Institute of Technology & Research, Wagholi, Pune

Department of Computer Engineering

Subject: LAB PRACTICE

Prepared By: PROF. PRIYA

SHARMA

Course Objectives:

- To learn and apply various search strategies for AI
- To Formalize and implement constraints in search problems
- To understand the concepts of Information Security / Augmented and Virtual Reality/Cloud Computing/Software Modeling and Architectures

Course Outcomes:

CO1: Design system using different informed search / uninformed search or heuristic approaches

CO2: Apply basic principles of AI in solutions that require problem solving, inference, perception, knowledge representation, and learning

CO3: Design and develop an expert system

CO4: Use tools and techniques in the area of Information Security

CO5: Use the knowledge of security for problem solving

CO6: Apply the concepts of Information Security to design and develop

Companion Course:

- Artificial Intelligence (310253)
- Elective II (310245)

HARDWARE AND SOFTWARE RERUIRED:

- Operating System recommended :- 64-bit Windows OS and Linux
- Programming tools recommended: -
- Information Security : - C/C++/Java/ Python

Course structure:

| Course Code | Course Name | Teaching Scheme (Hours/ week) | | | Examination Scheme and Marks | | | | | | Credit Scheme | | | |
|--------------|--|-------------------------------|-----------|----------|------------------------------|------------|------------|-----------|-----------|------------|---------------|-----------|----------|-----------|
| | | Lecture | Practical | Tutorial | Mid-Sem | End-Sem | Term work | Practical | Oral | Total | Lecture | Practical | Tutorial | Total |
| 310251 | Data Science and Big Data Analytics | 03 | - | - | 30 | 70 | - | - | - | 100 | 03 | | | 03 |
| 310252 | Web Technology | 03 | - | - | 30 | 70 | - | - | - | 100 | 03 | | | 03 |
| 310253 | Artificial Intelligence | 03 | - | - | 30 | 70 | - | - | - | 100 | 03 | | | 03 |
| 310254 | Elective II | 03 | - | - | 30 | 70 | - | - | - | 100 | 03 | | | 03 |
| 310255 | Internship | - | 04 | - | - | - | 100 | - | - | 100 | | 04 | | 04 |
| 310256 | Data Science and Big Data Analytics Laboratory | - | 04 | - | - | - | 25 | - | 50 | 75 | | 02 | | 02 |
| 310257 | Web Technology Laboratory | - | 04 | - | - | - | 25 | 50 | - | 75 | | 02 | | 02 |
| 310258 | Laboratory Practice II | - | 02 | - | - | - | 50 | - | - | 50 | | 01 | | 01 |
| 310259 | Audit Course 6 | | | | | | | | | | | | | |
| Total | | | | | | | | | | | 12 | 09 | - | 21 |
| Total | | 12 | 14 | - | 120 | 280 | 200 | 50 | 50 | 700 | 12 | 09 | - | 21 |

LIST OF PRACTICALS FOR AI

1. 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.
2. Implement A star Algorithm for any game search problem.
3. 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
4. Implement a solution for a Constraint Satisfaction Problem using Branch and Bound and Backtracking for n-queens problem or a graph colouring problem.
5. Develop an elementary chatbot for any suitable customer interaction application

6. Implement any one of the following Expert System

I. Information management

II. Hospitals and medical facilities

III. Help desks management

IV. Employee performance evaluation

V. Stock market trading

VI. Airline scheduling & cargo schedules

LIST OF PRACTICALS FOR IC

1. 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
2. Write a Java/C/C++/Python program to perform encryption and decryption using the method of Transposition technique
3. Write a Java/C/C++/Python program to implement DES algorithm
4. Write a Java/C/C++/Python program to implement AES Algorithm
5. Write a Java/C/C++/Python program to implement RSA algorithm.
6. Implement the Diffie-Hellman Key Exchange mechanism using HTML and JavaScript. Consider the end user as one of the parties (Alice) and the JavaScript application as other party (bob).
7. Calculate the message digest of a text using the MD5 algorithm in JAVA

Text Books:

- Stuart Russell and Peter Norvig, "Artificial Intelligence: A Modern Approach", Third edition, Pearson, 2003, ISBN :10: 0136042597
- Deepak Khemani, "A First Course in Artificial Intelligence", McGraw Hill Education(India), 2013, ISBN : 978-1-25-902998-1
- Elaine Rich, Kevin Knight and Nair, "Artificial Intelligence", TMH, ISBN-978-0-07-008770-5

Reference Books:

- Nilsson Nils J , "Artificial Intelligence: A new Synthesis, Morgan Kaufmann Publishers Inc. San Francisco, CA, ISBN: 978-1-55-860467-4

● Patrick Henry Winston, “Artificial Intelligence”, Addison-Wesley Publishing Company, ISBN: 0-201-53377-4

● Andries P. Engelbrecht-Computational Intelligence: An Introduction, 2nd Edition-Wiley India- ISBN:978-0-470-51250-0

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.

Definition:

- A Search strategy in which the highest layer of a decision tree is searched completely before proceeding to the next layer is called Breadth-first search (BFS).
- In this strategy, no viable solution is omitted and therefore guarantees that optimal solution is found.
- This strategy is often not feasible when the search space is large.
- Uses Queue(FIFO) data structure for its implementation

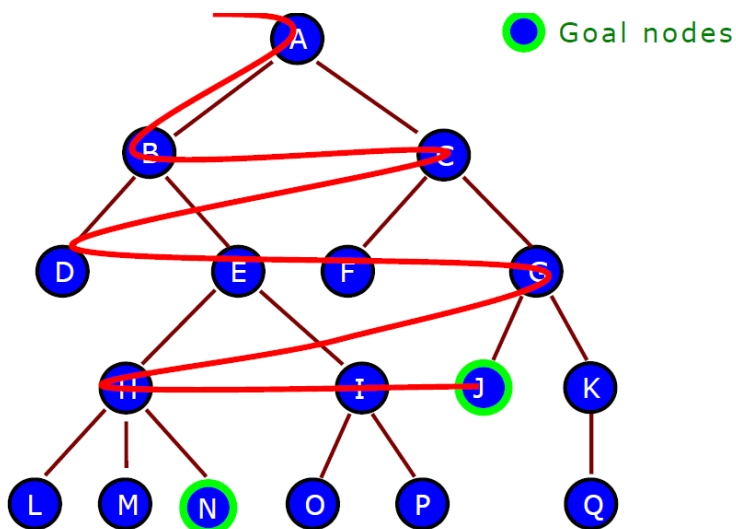


Fig. Breadth-first search (BFS)

Algorithm:

The steps involved in the BFS algorithm to explore a graph are given as follows

Step 1: SET STATUS = 1 (ready state) for each node in G

Step 2: Enqueue the starting node A and set its STATUS = 2 (waiting state)

Step 3: Repeat Steps 4 and 5 until QUEUE is empty

Step 4: Dequeue a node N. Process it and set its STATUS = 3 (processed state).

Step 5: Enqueue all the neighbors of N that are in the ready state (whose STATUS = 1) and set their STATUS = 2 (waiting state)

[END OF

LOOP]Step 6:

EXIT

Advantages:

- Guaranteed to find a solution if exists(i.e Complete)
- Guaranteed to provide optimal solution(i.e Optimal)
- This is guaranteed by the fact that longer paths are never explored until a shorter ones have already been examined.

Disadvantages:

- Requires more memory space as compared to DFS.

Code:

Import java.io.*;

Import java.util.*;

Public class BFSTraversal

{

Private int node; /* total number number of nodes in the graph */

Private LinkedList<Integer> adj[]; /* adjacency list */

Private Queue<Integer> que; /* maintaining a queue */

BFSTraversal(int v)

{

```
Node = v;
```

```
Adj = new LinkedList[node];
```

```
For (int i=0; i<v; i++)
```

```
{
```

```
    Adj[i] = new LinkedList<>();
```

```
}
```

```
Que = new LinkedList<Integer>();
```

```
}
```

```
Void insertEdge(int v,int w)
```

```
{
```

```
    Adj[v].add(w);  /* adding an edge to the adjacency list (edges are bidirectional in this example) */
```

```
}
```

```
Void BFS(int n)
```

```
{
```

```
    Boolean nodes[] = new boolean[node];    /* initialise boolean array for holding the data */
```

Int a = 0;

Nodes[n]=true;

Que.add(n); /* root node is added to the top of the queue */

While (que.size() != 0)

{

N = que.poll(); /* remove the top element of the queue */

System.out.print(n+" "); /* print the top element of the queue */

For (int I = 0; I < adj[n].size(); i++) /* iterate through the linked list and push all neighbours into queue */

{

A = adj[n].get(i);

If (!nodes[a]) /* only insert nodes into queue if they have not been explored already */

{

Nodes[a] = true;

Que.add(a);

}


```
}
```

```
}
```

```
}
```

```
Public static void main(String args[])
```

```
{
```

```
    BFSTraversal graph = new BFSTraversal(6);
```

```
    Graph.insertEdge(0, 1);
```

```
    Graph.insertEdge(0, 3);
```

```
    Graph.insertEdge(0, 4);
```

```
    Graph.insertEdge(4, 5);
```

```
    Graph.insertEdge(3, 5);
```

```
    Graph.insertEdge(1, 2);
```

```
    Graph.insertEdge(1, 0);
```

```
    Graph.insertEdge(2, 1);
```

```
    Graph.insertEdge(4, 1);
```

Graph.insertEdge(3, 1);

Graph.insertEdge(5, 4);

Graph.insertEdge(5, 3);

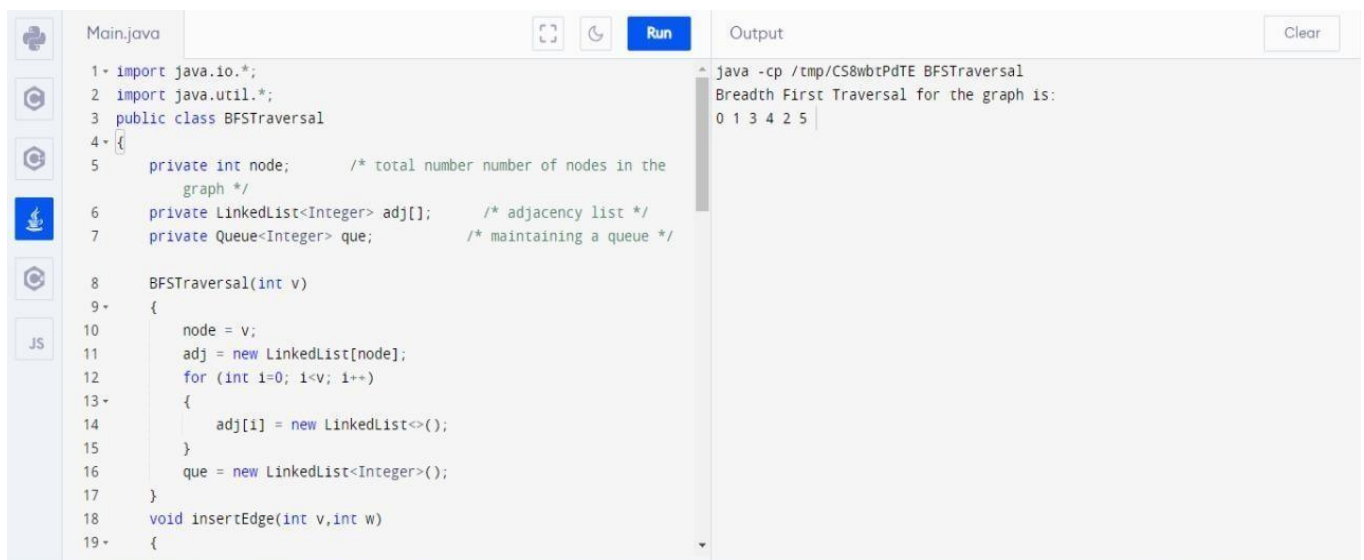
System.out.println("Breadth First Traversal for the graph is:");

Graph.BFS(0);

}

}

Output:



The screenshot shows an IDE with a file named 'Main.java'. The code defines a 'BFSTraversal' class with attributes for the number of nodes, an adjacency list, and a queue. It includes methods for initializing the graph and inserting edges. The output window shows the result of running the program: 'Breadth First Traversal for the graph is: 0 1 3 4 2 5'.

```
1- import java.io.*;
2 import java.util.*;
3 public class BFSTraversal
4 {
5     private int node;        /* total number number of nodes in the
6                               graph */
7     private LinkedList<Integer> adj[];    /* adjacency list */
8     private Queue<Integer> que;        /* maintaining a queue */
9
10    BFSTraversal(int v)
11    {
12        node = v;
13        adj = new LinkedList[node];
14        for (int i=0; i<v; i++)
15        {
16            adj[i] = new LinkedList<>();
17        }
18        que = new LinkedList<Integer>();
19    }
20    void insertEdge(int v,int w)
21    {
```

Output

```
java -cp /tmp/CS8wbtPdTE BFSTraversal
Breadth First Traversal for the graph is:
0 1 3 4 2 5
```

DFS (Depth first search)

- Uninformed search technique
- Works on present knowledge
- Used Stack (LIFO)
- Search till deepest node

- Incomplete
- Non optimal
- Time complexity
- $O(V + E)$
- $O(b^d)$

Algorithm:

1. Push the root node in stack
2. Loop until stack is empty
3. Peek the node of the stack
4. If the node has unvisited child nodes get the unvisited child node mark it has traverse and out it on stack.

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;
```

```
}
```

Output:

The screenshot shows a C++ IDE with a file named 'main.c'. The code implements a Depth-First Search (DFS) algorithm on a 7x7 grid. The grid 'A' is defined as follows:

```

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}
};

```

The DFS function is defined as:

```

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);
        }
    }
}

```

The output of the program is displayed in the 'Output' pane, showing the sequence of visited nodes: 0 1 2 3 4 5 6.

PRACTICAL NO:2

Title:

Implement A star Algorithm for any game search problem.

Defination:

- A* Algorithms are one of the best and most popular techniques used for path finding and graph traversals.
- A lot of games and web-based maps use this algorithm for finding the shortest path efficiently.
- It is essentially the best first search algorithm.

A* Algorithm works as-

- It maintains a tree of paths originating at the start node.
- It extends those paths one edge at a time.
- It continues until its termination criterion is satisfied.

A* Algorithm extends the path that minimizes the following function-

$$f(n) = g(n) + h(n)$$

Here,

‘n’ is the last node on the path

$g(n)$ is the cost of the path from start node to node ‘n’

$h(n)$ is a heuristic function that estimates cost of the cheapest path from node ‘n’ to the goal **node**

Algorithm:

Step1: Place the starting node in the OPEN list.

Step 2: Check if the OPEN list is empty or not, if the list is empty then return failure and stops.

Step 3: Select the node from the OPEN list which has the smallest value of evaluation function ($g+h$), if node n is goal node then return success and stop, otherwise

Step 4: Expand node n and generate all of its successors, and put n into the closed list. For each successor n', check whether n' is already in the OPEN or CLOSED list, if not then compute the evaluation function for n' and place it into the Open list.

Step 5: Else if node n' is already in OPEN and CLOSED, then it should be attached to the back pointer which reflects the lowest g(n') value.

Step 6: Return to Step 2.

Advantages:

A* search algorithm is the best algorithm than other search algorithms.

A* search algorithm is optimal and complete.

This algorithm can solve very complex problems.

Disadvantages:

It does not always produce the shortest path as it mostly based on heuristics and approximation. A*

search algorithm has some complexity issues.

The main drawback of A* is memory requirement as it keeps all generated nodes in the memory, so it is not practical for various large-scale problems.

Code:

```
#include <stdio.h>
```

```
#include
```

```
<stdlib.h>
```

```
#define STARTNODE 1
```

```
#define ENDNODE 2
```

```
#define BARRIER 3
```

```
typedef struct AStarNode
```

```
{
```

```
int s_x;//coordinates (required for final output path)
```

```
int s_y;
```

```
int s_g;//The distance from the starting point to this point (f can be obtained from g and h, here f is omitted, f = g + h)
```

```
int s_h;//distance from this point to the end point predicted by the heuristic function
```

```
int s_style;//Node type: start point, end point, obstacle
```

```
struct AStarNode * s_parent;//parent node

int s_is_in_closetable;//whether it is in the close table

int s_is_in_opentable;//whether it is in the open table

} AStarNode, * pAStarNode;
```

```
AStarNode map_maze [10] [10];//node array

pAStarNode open_table [100];//open table

pAStarNode close_table [100];//close table

int open_node_count;//Number of nodes in the open table

int close_node_count;//Number of nodes in the close table

pAStarNode path_stack [100];//stack to save the path

int top = -1;//top of stack
```

```
//swap two elements
```

```
//
```

```
void swap (int idx1, int idx2)
```

```
{
```

```
pAStarNode tmp = open_table [idx1];
```

```
open_table [idx1] = open_table [idx2];
```

```
open_table [idx2] = tmp;
```

```
}
```

```
//Heap adjustment
```

```
//
```

```
void adjust_heap (int/* i */nIndex)
```

```
{
```

```
int curr = nIndex;
```

```
int child = curr * 2 + 1;//get the idx of the left child (the subscript starts from 0, all children are curr * 2
```

```
+ 1) int parent = (curr-1)/2;//get parent idx
```

```

if (nIndex < 0 || nIndex >= open_node_count)
{
return;
}

//Adjust downwards (to compare left and right children with cuur parent)
//
while (child < open_node_count)
{
//Small root heap is the parent value is less than the child value
//
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; //Determine the size of the left and right children
}

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); //swap node
curr = child; //judge the current child node
child = curr * 2 + 1; //judge the left child
again
}
}

if (curr != nIndex)

```



```

{
return;
}

//Adjust upwards (only need to compare cuur child and parent)
//
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;
}
}
}

//Determine whether the neighbor can enter the open table
//
void insert_to_opentable (int x, int y, pAStarNode curr_node, pAStarNode end_node, int w)
{
int i;

if (map_maze [x] [y] .s_style != BARRIER)//not an obstacle
{
if (! map_maze [x] [y] .s_is_in_closetable)//Not in the closed table

```

```

{
if (map_maze [x] [y] .s_is_in_opentable)//in the open table
{
//Need to judge whether it is a more optimized path
//
if (map_maze [x] [y] .s_g > curr_node-> s_g + w)//if more optimized
{
map_maze [x] [y] .s_g = curr_node-> s_g + w;
map_maze [x] [y] .s_parent = curr_node;

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);//Adjustment point below
}
}
else//not in open
{
map_maze [x] [y] .s_g = curr_node-> s_g + w;
map_maze [x] [y] .s_h = abs (end_node-> s_x-x) + abs (end_node-> s_y-y);
map_maze [x] [y] .s_parent = curr_node;
map_maze [x] [y] .s_is_in_opentable = 1;
open_table [open_node_count ++] = & (map_maze [x] [y]);
}
}

```

```
}
```

```
}
```

```
//Find neighbors
```

```
//Find 8 neighbors up, down, left and right
```

```
//
```

```
void get_neighbors (pAStarNode curr_node, pAStarNode end_node)
```

```
{
```

```
int x = curr_node->
```

```
s_x;int y = curr_node->
```

```
> s_y;
```

```
//Let's deal with 8 neighbors below!
```

```
//
```

```
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);
}
}

int main ()
{
//The definition of the map array
//
AStarNode * start_node;//start point

```

```

AStarNode * end_node;//End point

AStarNode * curr_node;//current point

int is_found;//Whether to find the path

int maze [] [10] = { //just assign to map_maze for good

{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},

};

int i, j, x;


//Prepare below

//

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

{

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

{

map_maze [i] [j] .s_g =

0;map_maze [i] [j] .s_h

= 0;

map_maze [i] [j] .s_is_in_closetable = 0;

map_maze [i] [j] .s_is_in_opentable = 0;

map_maze [i] [j] .s_style = maze [i] [j];

map_maze [i] [j] .s_x = i;

map_maze [i] [j] .s_y = j;

```

```
map_maze [i] [j] .s_parent = NULL;
```

```
if (map_maze [i] [j] .s_style == STARTNODE) //starting point
```

```
{
```

```
start_node = & (map_maze [i] [j]);
```

```
}
```

```
else if (map_maze [i] [j] .s_style == ENDNODE) //end point
```

```
{
```

```
end_node = & (map_maze [i] [j]);
```

```
}
```

```
printf ("% d", maze [i] [j]);
```

```
}
```

```
printf ("\n");
```

```
}
```

```
//The following uses the A * algorithm to get the path
```

```
//
```

```
open_table [open_node_count ++] = start_node;//add the start point to the open table
```

```
start_node-> s_is_in_opentable = 1;//join the open table
```

```
start_node-> s_g = 0;
```

```
start_node-> s_h = abs (end_node-> s_x-start_node-> s_x) + 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)
```

```
{
```

```
printf ("Start point == End point!\n");
```

```
return 0;
```

```
}
```

```
is_found = 0;
```

```
while (1)
```

```
{
```

```
//for test
```

```
//
```

```
/* for (x = 0; x < open_node_count; ++ x)
```

```
{
```

```
printf ("% d,% d):% d", open_table [x]-> s_x, open_table [x]-> s_y, open_table [x]-> s_g + open_table [x]-> s_h);
```

```
}
```

```
printf ("\n\n");
```

```
*/
```

```
curr_node = open_table [0];//The first point of the open table must be the point with the smallest f value  
(obtained by heap sorting)
```

```
open_table [0] = open_table [-open_node_count];//Put the last point on the first point, and then adjust the heap
```

```
adjust_heap (0);//adjust the heap
```

```
close_table [close_node_count ++] = curr_node;//add the current point to the close table
```

```
curr_node-> s_is_in_closetable = 1;//Already in the close table
```

```
if (curr_node-> s_x == end_node-> s_x && curr_node-> s_y == end_node-> s_y)//The end point is in close, end
```

```
{
```

```
is_found =
```

```
1;break;
```

```
}
```

```
get_neighbors (curr_node, end_node);//Deal with neighbors
```

```
if (open_node_count == 0)//no path arrives
```

```

{
is_found =
0;break;
}
}

if (is_found)
{
curr_node = end_node;

while (curr_node)
{
path_stack [++ top] = curr_node;
curr_node = curr_node-> s_parent;
}

while (top >= 0)//Here is the output path ~
{
if (top> 0)
{
printf ("% d,% d)->", path_stack [top]-> s_x, path_stack [top]-> s_y);
}
else
{
printf ("% d,% d)", path_stack [top]-> s_x, path_stack [top]-> s_y);
}
}
}
else
{

```



```
printf ("Why did you find the path");
```

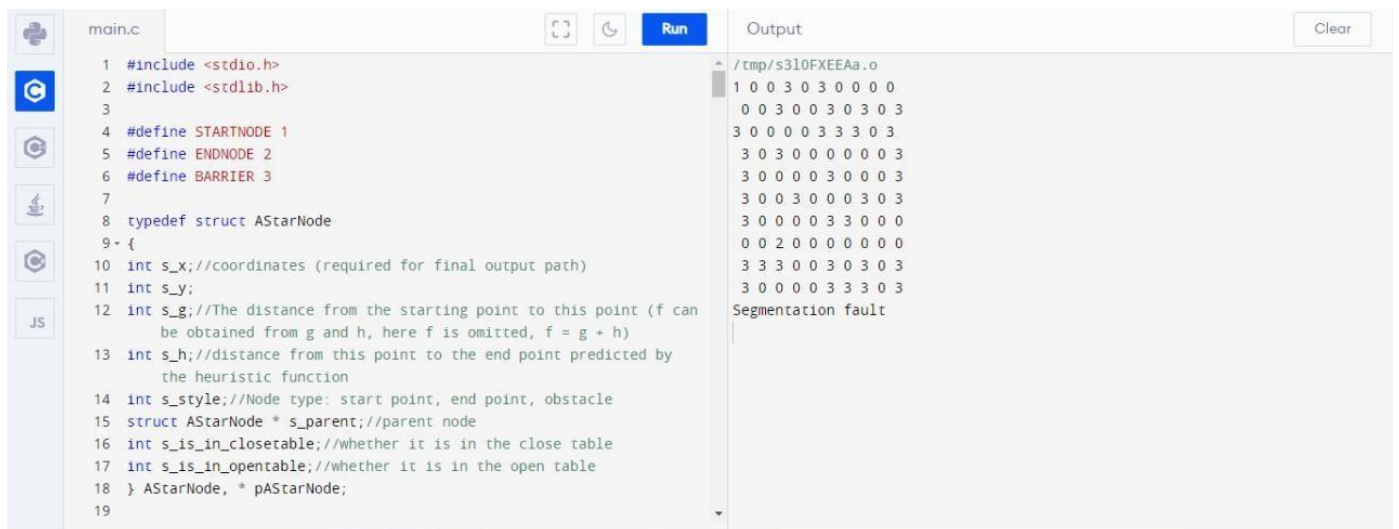
```
}
```

```
puts ("");
```

```
return 0;
```

```
}
```

Output:



The screenshot shows a code editor with a file named 'main.c'. The code is a C program that defines constants for start, end, and barrier nodes, and defines a struct for AStarNode. The output window shows a 10x10 grid of 0s and 3s, representing a pathfinding map. The path is highlighted by 0s, and obstacles are represented by 3s. The output ends with a 'Segmentation fault' message.

```
main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define STARTNODE 1
5 #define ENDNODE 2
6 #define BARRIER 3
7
8 typedef struct AStarNode
9 {
10 int s_x; //coordinates (required for final output path)
11 int s_y;
12 int s_g; //The distance from the starting point to this point (f can
    be obtained from g and h, here f is omitted, f = g + h)
13 int s_h; //distance from this point to the end point predicted by
    the heuristic function
14 int s_style; //Node type: start point, end point, obstacle
15 struct AStarNode * s_parent; //parent node
16 int s_is_in_closetable; //whether it is in the close table
17 int s_is_in_opentable; //whether it is in the open table
18 } AStarNode, * pAStarNode;
19
```

Output

```
/tmp/s3l0FXEEAa.o
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
Segmentation fault
```

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

Theory:

- Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach.
- This algorithm treats the graph as a forest and every node it has as an individual tree.
- A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties

Algorithm:

- Kruskal's Algorithm is a famous greedy algorithm.
- It is used for finding the Minimum Spanning Tree (MST) of a given graph.
- To apply Kruskal's algorithm, the given graph must be weighted, connected and undirected.

Steps:

- Sort all the edges from low weight to high weight.
- Take the edge with the lowest weight and use it to connect the vertices of graph.
- If adding an edge creates a cycle, then reject that edge and go for the next least weight edge.
- Keep adding edges until all the vertices are connected and a Minimum Spanning Tree (MST) is obtained.

Code :

```
#include<stdio.h>

#include<conio.h>

>

#include<stdlib.h>

>

int i,j,k,a,b,u,v,n,ne=1;

int min,mincost=0,cost[9][9],parent[9];

int find(int);

int

uni(int,int);

void main()

{

    printf("\n\tImplementation of Kruskal's

    algorithm\n");printf("\nEnter the no. of vertices:");

    scanf("%d",&n);

    printf("\nEnter the cost adjacency matrix:\n");

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

    {

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

        {

            scanf("%d",&cost[i][j

            ]);if(cost[i][j]==0)

                cost[i][j]=999;

        }

    }

    printf("The edges of Minimum Cost Spanning Tree are\n");

    while(ne < n)

    {

        for(i=1,min=999;i<=n;i++)

        {
```

```

        for(j=1;j <= n;j++)
        {
            if(cost[i][j] < min)
            {
                min=cost[i][j]
                ;a=u=i;
                b=v=j;
            }
        }
    }
    u=find(u);
    v=find(v);
    if(uni(u,v)
    )
    {
        printf("%d edge (%d,%d)=%d\n",ne++,a,b,min);
        mincost +=min;
    }
    cost[a][b]=cost[b][a]=999;
}

printf("\n\tMinimum cost =
%d\n",mincost);getch();
}

int find(int i)
{
    while(parent[i]
    )i=parent[i];
    return i;
}

int uni(int i,int j)
{

```

```

    if(i!=j)
    {
        parent[j]=i
        ;return 1;
    }
    return 0;
}

```

Output:

```

Implementation of Kruskal's algorithm
Enter the no. of vertices:4
Enter the cost adjacency matrix:
2
1
3
6
13
5
9
10
15
7
2
17
12
19
3
4
The edges of Minimum Cost Spanning Tree are
1 edge (1,2) =1
2 edge (1,3) =3
3 edge (4,3) =3
Minimum cost = 7
...Program finished with exit code 0
Press ENTER to exit console.

```

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

Theory:

What is graph coloring problem?

Graph coloring problem involves assigning colors to certain elements of a graph subject to certain restrictions and constraints. This has found applications in numerous fields in computer science. For example:

Sudoku:

This game is a variation of Graph coloring problem where every cell denotes a node (or vertex) and there exists an edge between two nodes if the nodes are in same row or same column or same block.

Using Backtracking:

By using the backtracking method, the main idea is to assign colors one by one to different vertices right from the first vertex (vertex 0).

Before color assignment, check if the adjacent vertices have same or different color by considering already assigned colors to the adjacent vertices.

If the color assignment does not violate any constraints, then we mark that color as part of the result. If color assignment is not possible then backtrack and return false

Algorithm:

1. There exists no efficient algorithm for coloring a graph with minimum number of colors.
2. Graph Coloring is a NP complete problem.

Steps:

1. Color first vertex with the first color.
2. Now, consider the remaining (V-1) vertices one by one and do the following-
3. Color the currently picked vertex with the lowest numbered color if it has not been used to color any of its adjacent vertices.
4. If it has been used, then choose the next least numbered color.
5. If all the previously used colors have been used, then assign a new color to the currently picked vertex.

Complexity:

- Time Complexity: $O(m^V)$.
- Space Complexity: $O(V)$ which is for storing the output array.

Code:

N = 8

""" A utility function to print solution

"""def printSolution(board):

for i in range(N):

for j in range(N):

print(board[i][j], end = " ")

print()

""" A Optimized function to check if

a queen can be placed on board[row][col]

"""def isSafe(row, col, slashCode,

backslashCode,

rowLookup, slashCodeLookup,

backslashCodeLookup

):if (slashCodeLookup[slashCode[row][col]] or

backslashCodeLookup[backslashCode[row][col]] or

rowLookup[row]):

return

False
 return True

""" A recursive utility

function to solve N Queen

problem """

def solveNQueensUtil(board, col, slashCode, backslashCode,

rowLookup,

slashCodeLookup,

backslashCodeLookup):

""" base case: If all queens are

placed then return True """

if(col >= N):

```

        return

    for i in
    range(N):
        if(isSafe(i, col, slashCode, backslashCode,
                    rowLookup,
                    slashCodeLookup,
                    backslashCodeLookup)):

            """ Place this queen in board[i][col] """

            board[i][col] = 1

            rowLookup[i] = True

            slashCodeLookup[slashCode[i][col]] = True

            backslashCodeLookup[backslashCode[i][col]] =
            True

            """ recur to place rest of the queens """

            if(solveNQueensUtil(board, col + 1,
                                slashCode, backslashCode,
                                rowLookup,
                                slashCodeLookup,
                                backslashCodeLookup)):

                return True

            """ If placing queen in board[i][col]
            doesn't lead to a solution,then backtrack """

            """ Remove queen from board[i][col] """

            board[i][col] = 0

            rowLookup[i] = False

            slashCodeLookup[slashCode[i][col]] = False

            backslashCodeLookup[backslashCode[i][col]] =
            False

```


""" If queen can not be place in any row in
this column col then return False """

return False

""" This function solves the N Queen problem using
Branch or Bound. It mainly uses
solveNQueensUtil() to solve the problem. It returns
False if queens
cannot be placed, otherwise return True or
prints placement of queens in the form of 1s.
Please note that there may be more than one
solutions, this function prints one of the
feasible solutions. """

def solveNQueens():

board = [[0 for i in range(N)]
 for j in range(N)]

helper matrices

slashCode = [[0 for i in range(N)]
 for j in
range(N)]
backslashCode = [[0 for i in
range(N)]
 for j in range(N)]

arrays to tell us which rows are occupied

rowLookup = [False] * N

keep two arrays to tell us

which diagonals are occupied

x = 2 * N - 1

slashCodeLookup = [False] * x

backslashCodeLookup = [False] * x

```
# initialize helper matrices
```

```
for rr in range(N):
```

```
    for cc in range(N):
```

```
        slashCode[rr][cc] = rr + cc
```

```
        backslashCode[rr][cc] = rr - cc + 7
```

```
if(solveNQueensUtil(board, 0, slashCode, backslashCode,
```

```
                    rowLookup, slashCodeLookup,
```

```
                    backslashCodeLookup) ==
```

```
False):
```

```
    print("Solution does not exist")
```

```
    return False
```

```
# solution found
```

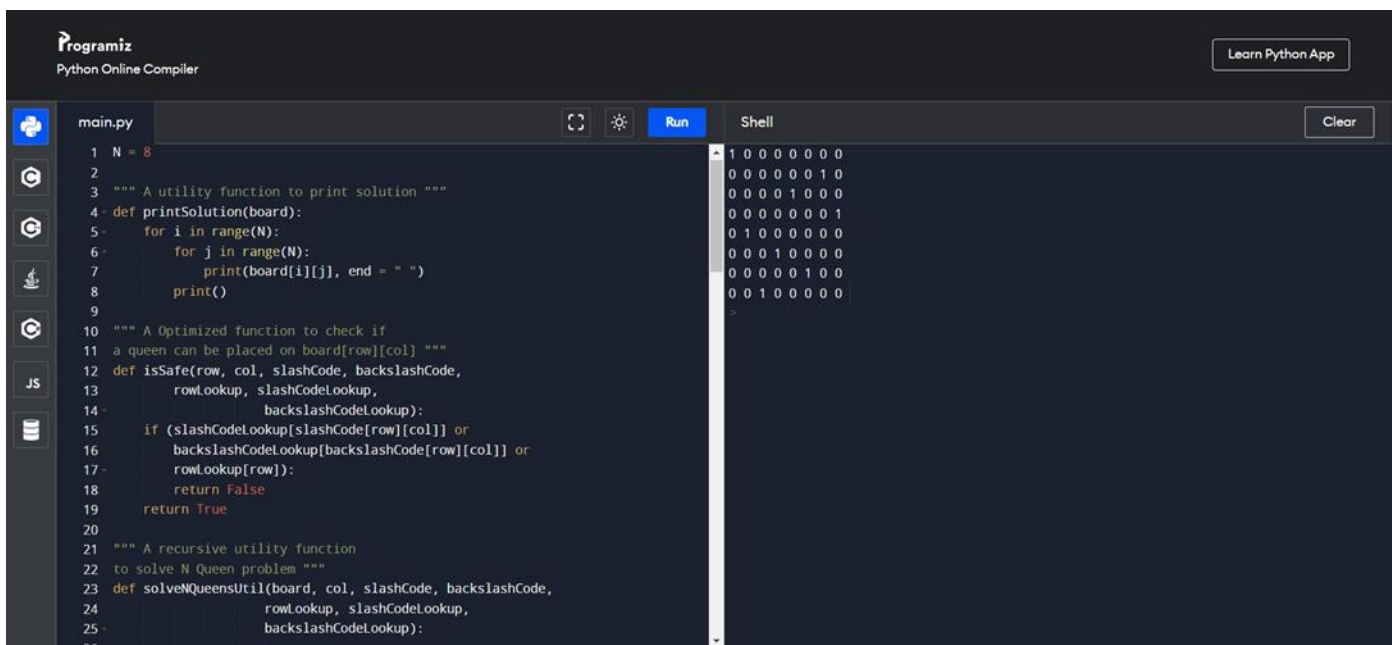
```
printSolution(board
```

```
)return True
```

```
# Driver Cde
```

```
solveNQueens()
```

Output:



```
Programiz
Python Online Compiler
Learn Python App

main.py
1 N = 8
2
3 """ A utility function to print solution """
4 def printSolution(board):
5     for i in range(N):
6         for j in range(N):
7             print(board[i][j], end = " ")
8         print()
9
10 """ A Optimized function to check if
11 a queen can be placed on board[row][col] """
12 def isSafe(row, col, slashCode, backslashCode,
13            rowLookup, slashCodeLookup,
14            backslashCodeLookup):
15     if (slashCodeLookup[slashCode[row][col]] or
16         backslashCodeLookup[backslashCode[row][col]] or
17         rowLookup[row]):
18         return False
19     return True
20
21 """ A recursive utility function
22 to solve N Queen problem """
23 def solveNQueensUtil(board, col, slashCode, backslashCode,
24                      rowLookup, slashCodeLookup,
25                      backslashCodeLookup):
26     ~

Shell
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```

PRACTICAL NO:5

Title: Develop an elementary chatbot for any suitable customer interaction

application. Theory:

What is a chatbot?

A chatbot is a computer program designed to have a conversation with human beings over the internet. It's also known as conversational agents, which communicate and collaborate with human users, through text messaging, in order to accomplish a specific task. Basically, there are two types of chatbots. The one that uses Artificial Intelligence, and another one is based on multiple choice scripts. Both types of chatbots aim to create a more personalized content experience for the users, whether that's while watching a video, reading articles or buying new shoes. These Chatbots hold the promise of being the next generation of technology that people use to interact online with business enterprises. These Chatbots offer a lot of advantages, one of which is that, because Chatbots communicate using a natural language, users don't need to learn yet another new website interface, to get comfortable with the unavoidable quirks. Chatbots are capable of interpreting human speech, and deciding which information is being sought. Artificial intelligence is getting smarter each day, and brands that are integrating Chatbots with the artificial intelligence, can deliver one-to-one individualized experiences to consumers.

Why chatbot?

Chatbots can be useful in many aspects of the customer experience, including providing customer service, presenting product recommendations and engaging customers through targeted marketing campaigns. If a customer has an issue with a product, she can connect with a chatbot to explain the situation and the chatbot can input that information to provide a recommendation of how to fix the product. On the recommendation side, chatbots can be used to share popular products with customers that they might find useful and can act as a sort of personal shopper or concierge service to find the perfect gift, meal or night out for a customer with just a few basic questions. Brands are also using chatbots to connect their customers with thought leaders and add personality to their products. In all cases, brands seem to be having great success and experiencing increased engagement and revenue. Chatbots are easy to use and many customers prefer them over calling a representative on the phone because it tends to be faster and less invasive. They can also save money for companies and are easy to set up. Chatbots are relatively new and most companies haven't implemented them yet, it's only natural that users are interested in them. Hence, people want to discover what chatbots can and cannot do. The number of businesses using chatbots has grown exponentially. Chatbots have increased from 30,000 in 2016 to over 100,000 today. Every major company has announced their own chatbot and 60% of the youth population uses them daily. These statistics prove that chatbots are the new-gen tech. No more waiting for the right time to incorporate them into your business. The time is now. By the year 2020, nearly 80% of businesses will have their own chatbot.

Benefits of chatbots?

Chatbots are being made to ease the pain that the industries are facing today. The purpose of chat bots is to support and scale business teams in their relations with customers. Chatbots may sound like a futuristic notion, but according to Global Web Index statistics, it is said that 75% of internet users are adopting one or more messenger platforms. Although research shows us that each user makes an average of 24 apps a month, wherein 80% of the time would be in just 5 apps. This means you can hardly shoot ahead with an app, but you still have high chances to integrate your chatbot with one of these platforms.

Now let's go through some of the benefits that chatbots provide:

1. Available 24*7:

I'm sure most of you have experienced listening to the boring music playing while you're kept on hold by a customer care agent. On an average people spend 7 minutes until they are assigned to an agent. Gone are the days of waiting for the next available operative. Bots are replacing live chat and other forms of contact such as emails and phone calls. Since chatbots are basically virtual robots they never get tired and continue to obey your command. They will continue to operate every day throughout the year without requiring to take a break. This improves your customer satisfaction and helps you rank highly in your sector.

Handling Customers:

We humans are restricted to the number of things we can do at the same time. A study suggests that humans can only concentrate on 3–4 things at the same time. If it goes beyond that you are bound to meet errors. Chatbots on the other hand can simultaneously have conversations with thousands of people. No matter what time of the day it is or how many people are contacting you, every single one of them will be answered instantly. Companies like Taco Bell and Domino's are already using chatbots to arrange delivery of parcels.

Helps you Save Money:

If you are a business owner you are bound to have a lot of employees who need to be paid for the work they do. And these expenses just keep adding up as business grows. Chatbots are a one time investment which helps businesses reduce down on staff required. You could integrate a customer support chatbot in your business to cater to simple queries of customers and pass on only the complex queries to customer support agents.

Provides 100% satisfaction to customers:

Humans react to others based on their mood and emotions. If an agent is having a good attitude or is in a good mood he will most probably talk to customers in a good way. In contrast to this the customer will not be satisfied. Whereas chatbots are bound by some rules and obey them as long as they're programmed to. They always treat a customer in the most polite and perfect way no matter how rough the person is. Also, in the travel and hospitality industry where travelers do not speak the same language, a bot can be trained to communicate in the language of the traveler. Automation of repetitive work:

Let's be honest, no one likes doing the same work again and again over a brief period of time. In the case of humans, such tasks are prone to errors. Chatbots now help automate tasks which are to be done frequently and at the right time. Also, now there are numerous slack bots which automate repetitive tasks. This helps people save time and increase productivity. For example, there are new items bought from your eCommerce site or there is a bug reported then it sends a short summary to a slack channel.

Personal Assistant:

People could use Bots as a fashion advisor for clothing recommendations, or ask trading tips from a finance bot, suggest places to visit from a travel bot and so forth. This would help the users get a more personal touch from the chatbot. Also, the chatbot will remember all your choices and provide you with relevant choices the next time you visit it. How can you drive revenue for you? Below we have compiled reasons why chatbots are important for your business and how can they help in increasing revenues:

Higher user customer engagement

Most businesses these days have a web presence. But with being on the internet, boundaries of day and night, availability and unavailability have changed, so have user expectations. This is probably the biggest reason to use them. Bots give the user an interactive experience. It makes customers feel they are working with someone to help resolve their issue. If done right, bots can help customers find what they are looking for and make them more likely to return.

Customer Engagement

- Clearance Sale : Notify users about on-going clearance sale of products relevant to the users at their nearest outlets.
- Product Finder : Enable consultative selling without the need of a call center
- It offer Notification : Notify users about offers, product launches on products/ services they've shown interest in, and products that's back in stock

Mobile-ready and immediate availability

Along with a web presence, it has also become increasingly important for brands to have a mobile presence - mobile apps, mobile-optimized websites. Considering how chat has been around on the mobile for ages, most chatbot implementations don't need you to work on tweaking their UI, they are ready to implement and so available to your customers immediately. You might argue that you have an app for that. Having an app for your brand is great, but having users discover that app, download it and use it to stay engaged is not an easy deal. Instead, implementing a chatbot - which works on the mobile browser or a messaging-app which the user regularly uses - makes it all the more reason for a customer to be engaged with the brand

It can drive sales

Chatbots can be intelligent. Depending on a user's preferences or purchases, it can send products to customers which are more likely to convert into sales. Or it can send coupons to users for in-store purchases/discounts. Bots can also be used to link the user to your eCommerce site/app so they can buy the product directly from the convenience of their phones

Sell Intelligently

- Product Recommendations : Push proactive recommendations to users based on their preferences and search and order history.
- Enable order booking over chat.

Minimal cost - Maximum return

The best part about bots is they are cheap. Chatbot provides the necessary infrastructure and APIs for creating these bots. They require minimal maintenance and since it is automated, there is no labor-intensive work that goes in there.

Customer Service

- Track Order : Keep users up to date with order status. Schedule or reschedule delivery to a provided address or request to pick it up at any other Best Buy outlet.
- Stock outs : Notify users when desired product is available and place order over a chat.
- Returns and Replacements : No waiting time to reach customer care. Customers can instantly place a request to replace or return an order.
- Seek Reviews : Reach out to users to seek reviews on the products recently bought

Gift Recommendations

- Recommend relevant gifting options to users, accessing calendar events and understanding the likes and style of beneficiary.
- Opportunity to upsell gift cards for the users for every occasion.

Application across Industries

According to a new survey, 80% of businesses want to integrate chatbots in their business model by 2020. So which industries can reap the greatest benefits by implementing consumer-facing chatbots? According to a chatbot, these major areas of direct-to-consumer engagement are prime:

Chatbots in Restaurant and Retail Industries

Famous restaurant chains like Burger King and Taco Bell have introduced their Chatbots to stand out from competitors of the Industry as well as treat their customers quickly. Customers of these restaurants are greeted by the resident Chatbots, and are offered the menu options- like a counter order, the Buyer chooses their pickup location, pays, and gets told when they can head over to grab their food. Chatbots also work to accept table reservations, take special requests and go the extra step to make the evening special for your guests. Chatbots are not only good for the restaurant staff in reducing work and pain but can provide a better user experience for the customers.

Chatbots in Hospitality and Travel

For hoteliers, automation has been held up as a solution for all difficulties related to productivity issues, labor costs, a way to ensure consistency, streamlined production processes across the system. Accurate and immediate delivery of information to customers is a major factor in running a successful online Business, especially in the price sensitive and competitive Travel and Hospitality industry. Chatbots particularly have gotten a lot of attention from the hospitality industry in recent months. Chatbots can help hotels in a number of areas, including time management, guest services and cost reduction. They can assist guests with elementary questions and requests. Thus, freeing up hotel staff to devote more of their time and attention to time-sensitive, critical, and complicated tasks. They are often more cost effective and faster than their human counterparts. They can be programmed to speak to guests in different languages, making it easier for the guests to speak in their local language to communicate.

Chatbots in Health Industry

Chatbots are a much better fit for patient engagement than Standalone apps. Through These Health-Bots, users can ask health related questions and receive immediate responses. These responses are either original or based on responses to similar questions in the database. The impersonal nature of a bot could act as a benefit in certain situations, where an actual Doctor is not needed. Chatbots ease the access to healthcare and industry has favorable chances to serve their

customers with personalized health tips. It can be a good example of the success of Chatbots and Service Industry combo.

Chatbots in E-Commerce

Mobile messengers- connected with Chatbots and the E-commerce business can open a new channel for selling the products online. E-commerce Shopping destination “Spring” was the early adopter. E-commerce future is where brands have their own Chat Bots which can interact with their customers through their apps.

Chatbots in Fashion Industry

Chatbots, AI and Machine Learning pave a new domain of possibilities in the Fashion industry, from Data Analytics to Personal Chatbot Stylists. Fashion is such an industry where luxury goods can only be bought in a few physical boutiques and one to one customer service is essential. The Internet changed this dramatically, by giving the customers a seamless but a very impersonal experience of shopping. This particular problem can be solved by Chatbots. Customers can be treated personally with bots, which can exchange messages, give required suggestions and information. Famous fashion brands like Burberry, Tommy Hilfiger have recently launched Chatbots for the London and New York Fashion Week respectively. Sephora a famous cosmetics brand and H&M– a fashion clothing brand have also launched their Chatbots.

Chatbots in Finance

Chatbots have already stepped into the Finance Industry. Chatbots can be programmed to assist the customers as Financial Advisor, Expense Saving Bot, Banking Bots, Taxbots, etc. Banks and Fintech have ample opportunities in developing bots for reducing their costs as well as human errors. Chatbots can work for customer’s convenience, managing multiple accounts, directly checking their bank balance and expenses on particular things. Further about Finance and Chatbots have been discussed in our earlier blog: Chatbots as your Personal Finance Assistant.

Chatbots in Fitness Industry

Chat based health and fitness companies using Chatbot, to help their customers get personalized health and fitness tips. Tech based fitness companies can have a huge opportunity by developing their own Chatbots offering a huge customer base with personalized services. Engage with your fans like never before with news, highlights, game-day info, roster and more. Chatbots and Service Industry together have a wide range of opportunities and small to big companies of all sizes are using chatbots to reduce their work and help their customers better.

Chatbot in Celebrity:

With a chatbot you can now have one-on-one conversation with millions of fans.

Chatbot in Marketing SMS Marketing

- Why promote just a coupon code that the customer does not know how to use?
- Improve conversions from your existing SMS campaigns.
- Talk to your customers when they want to using “Talk to an Agent” feature.

Email Marketing

- So your eMail has made a solid elevator pitch about your product.
- As a next step, is making customers fill an online form the most exciting way to engage with your customers?
- It's time to rethink the landing page.
- Instantly engage in a conversation with your customers.
- Address their concerns and queries

Social Media Triage

- How effectively are you addressing the negative sentiment around your brand on social media?
- Addressing queries instantly and effectively can convert even an angry customer into a loyal fan.
- Leverage a chatbot as your first response strategy and comfort that customer.

Process:

Stage #1: Chatty Bot welcomes you. Teach your assistant to introduce itself in

the console. Stage #2: Print your name Introduce yourself to the bot.

Stage #3: Guess the age Use your knowledge of strings and numbers to make the assistant guess your age.

Stage #4: Learning numbers Your assistant is old enough to learn how to count. And you are experienced enough to apply for a loop at this stage!

Stage #5: Multiple Choice At this point, the assistant will be able to check your knowledge and ask multiple choice questions. Add some functions to your code and make the stage even better.

How To Run The Project?

To run this project, you must have installed [Python](#) on your PC. After downloading the project, follow the steps below:

Step1: Extract/Unzip the file

Step2: Go inside the project folder, open cmd then type bot.py and enter to start the system. OR Step2: Simply, double-click the bot.py file and you are ready to go.

PRACTICAL NO:6

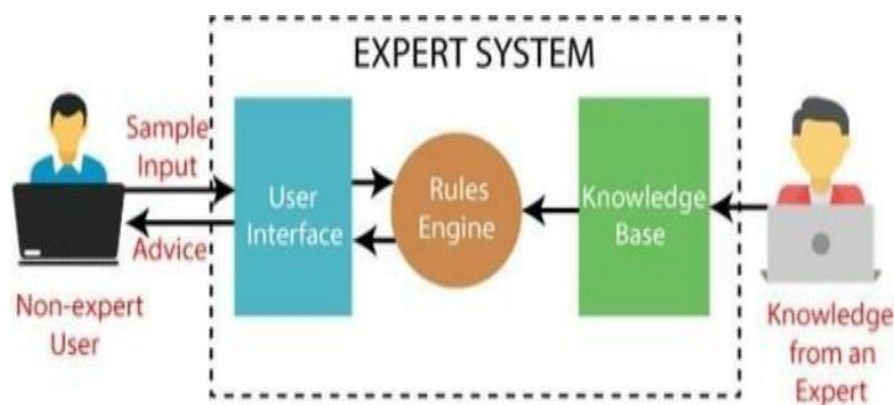
Title: Implement any one of the following Expert System

1. Information management
2. Hospitals and medical facilities
3. Help desks management
4. Employee performance evaluation
5. Stock market trading
6. Airline scheduling and cargo schedules

Theory:

What is an Expert System?

An expert system is a computer program that is designed to solve complex problems and to provide decision-making ability like a human expert. It performs this by extracting knowledge from its knowledge base using the reasoning and inference rules according to the user queries. The expert system is a part of AI, and the first ES was developed in the year 1970, which was the first successful approach of artificial intelligence. It solves the most complex issue as an expert by extracting the knowledge stored in its knowledge base. The system helps in decision making for complex problems using **both facts and heuristics like a human expert**. It is called so because it contains the expert knowledge of a specific domain and can solve any complex problem of that particular domain. These systems are designed for a specific domain, such as **medicine, science**, etc. The performance of an expert system is based on the expert's knowledge stored in its knowledge base. The more knowledge stored in the KB, the more that system improves its performance. One of the common examples of an ES is a suggestion of spelling errors while typing in the Google search box. Below is the block diagram that represents the working of an expert system:



Below are some popular examples of the Expert System:

- o **DENDRAL:** It was an artificial intelligence project that was made as a chemical analysis expert system. It was used in organic chemistry to detect unknown organic molecules with the help of

their mass spectra and knowledge base of chemistry.

- o **MYCIN:** It was one of the earliest backward chaining expert systems that was designed to find the bacteria causing infections like bacteraemia and meningitis. It was also used for the recommendation of antibiotics and the diagnosis of blood clotting diseases.
- o **PXDES:** It is an expert system that is used to determine the type and level of lung cancer. To determine the disease, it takes a picture from the upper body, which looks like a shadow. This shadow identifies the type and degree of harm.
- o **CaDeT:** The CaDet expert system is a diagnostic support system that can detect cancer early

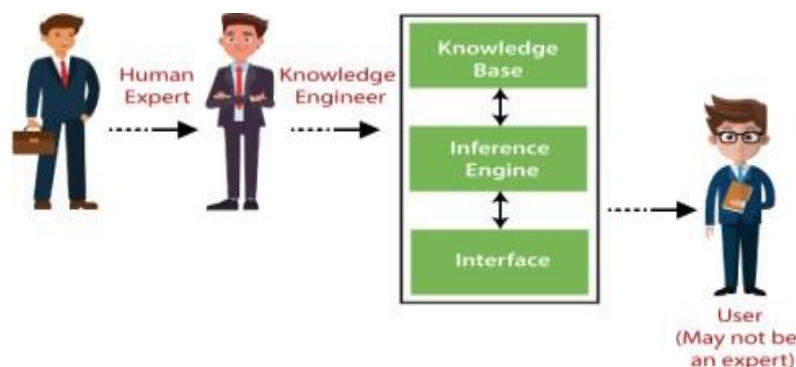
stages. **Characteristics of Expert System**

- o **High Performance:** The expert system provides high performance for solving any type of complex problem of a specific domain with high efficiency and accuracy.
- o **Understandable:** It responds in a way that can be easily understandable by the user. It can take input in human language and provide the output in the same way.
- o **Reliable:** It is much reliable for generating an efficient and accurate output.
- o **Highly responsive:** ES provides the result for any complex query within a very short period

of time. **Components of Expert System**

An expert system mainly consists of three components:

- o **User Interface**
- o **Inference Engine**
- o **Knowledge Base**



Participants in the development of Expert System

There are three primary participants in the building of Expert System:

1. **Expert:** The success of an ES much depends on the knowledge provided by human experts.

These experts are those persons who are specialized in that specific domain.

2. **Knowledge Engineer:** Knowledge engineer is the person who gathers the knowledge from the domain experts and then codifies that knowledge to the system according to the formalism.

3. **End-User:** This is a particular person or a group of people who may not be experts, and working on the expert system needs the solution or advice for his queries, which are complex.

Advantages of Expert System

- o These systems are highly reproducible.
- o They can be used for risky places where the human presence is not safe.
- o Error possibilities are less if the KB contains correct knowledge.
- o The performance of these systems remains steady as it is not affected by emotions, tension, or fatigue.
- o They provide a very high speed to respond to a particular query.

Limitations of Expert System

- o The response of the expert system may get wrong if the knowledge base contains the wrong information.
- o Like a human being, it cannot produce a creative output for different scenarios.
- o Its maintenance and development costs are very high.
- o Knowledge acquisition for designing is much difficult.
- o For each domain, we require a specific ES, which is one of the big limitations.
- o It cannot learn from itself and hence requires manual updates.

Applications of Expert System

o In designing and manufacturing domain

It can be broadly used for designing and manufacturing physical devices such as camera lenses and automobiles.

o In the knowledge domain

These systems are primarily used for publishing the relevant knowledge to the users. The two popular ES used for this domain is an advisor and a tax advisor.

o In the finance domain

In the finance industry, it is used to detect any type of possible fraud, suspicious activity, and advise bankers on whether they should provide loans for business or not.

o In the diagnosis and troubleshooting of devices

In medical diagnosis, the ES system is used, and it was the first area where these systems were used.

- **Planning and Scheduling**

The expert systems can also be used for planning and scheduling some particular tasks for achieving the goal of that task.

Source code

```
import random as r

print("Welcome to the Expert System!".center(60,'-')+"\n\n")

questions = [

    "The employee acts on all the feedback/inputs received from Managers",

    "The employee is willing to take on additional responsibilities",

    "The employee is able to attain work-life balance",

    "The employee is cognizant of the organizational goals and works towards them",

    "When working on a team project, the employee values everyone's contribution",

    "When attending team meetings, the employee is always on time",

    "The employee contributes suggestions to new initiatives or existing issues",

    "The employee is willing to contribute to proceedings for team meals or outings",

    "The employee is not averse to management and work-related changes",

]

def get_result(score):

    if score <= 1*MAX_QUESTIONS:

        print("You seem Happy with your work ! 🏆\nKeep it up!")

    elif score <= 2*MAX_QUESTIONS:

        print("You're liking your work 🏆🏆")
```

```

elif score <= 3*MAX_QUESTIONS:

    print("It looks like you're not happy with your job 🟡")

elif score <= 4*MAX_QUESTIONS:

    print("It looks like you hate your job 🟡")

else:

    print("It looks like you're about to quit. ")

    print("Try searching for new job on linkedin.com 🟡 ")

l = []

MAX_QUESTIONS = len(questions)

for i in range(MAX_QUESTIONS):

    l.append(0)

while 0 in l:

    qno = r.randint(0,MAX_QUESTIONS-1)

    if l[qno] == 0:

        print(questions[qno])

        exit = False

        while not exit:

            l[qno] = int(input('1.Always\t2.Sometimes\t3.yes\t4.Rarely\t5.Not at all\n Enter
Your choice:'))

            exit = True

            if l[qno] > 5 and l[qno] < 0:

                print("please Enter Valid Choice.")

            exit = False

get_result(sum(l))

```

Output:

```
input
-----Welcome to the Expert System!-----

The employee contributes suggestions to new initiatives or existing issues
1.Always      2.Sometimes    3.yes  4.Rarely    5.Not at all
Enter Your choice:1

When working on a team project, the employee values everyone's contribution
1.Always      2.Sometimes    3.yes  4.Rarely    5.Not at all
Enter Your choice:1

The employee is willing to contribute to proceedings for team meals or outings
1.Always      2.Sometimes    3.yes  4.Rarely    5.Not at all
Enter Your choice:1

The employee acts on all the feedback/inputs received from Managers
1.Always      2.Sometimes    3.yes  4.Rarely    5.Not at all
Enter Your choice:1

The employee is cognizant of the organizational goals and works towards them
1.Always      2.Sometimes    3.yes  4.Rarely    5.Not at all
Enter Your choice:1

The employee is willing to take on additional responsibilities
1.Always      2.Sometimes    3.yes  4.Rarely    5.Not at all
Enter Your choice:1

The employee is not averse to management and work-related changes
1.Always      2.Sometimes    3.yes  4.Rarely    5.Not at all
Enter Your choice:1

When attending team meetings, the employee is always on time
1.Always      2.Sometimes    3.yes  4.Rarely    5.Not at all
Enter Your choice:1

The employee is able to attain work-life balance
1.Always      2.Sometimes    3.yes  4.Rarely    5.Not at all
Enter Your choice:1

You seem Happy with your work ! 😊
Keep it up!

...Program finished with exit code 0
Press ENTER to exit console.
```

PRACTICAL NO 7

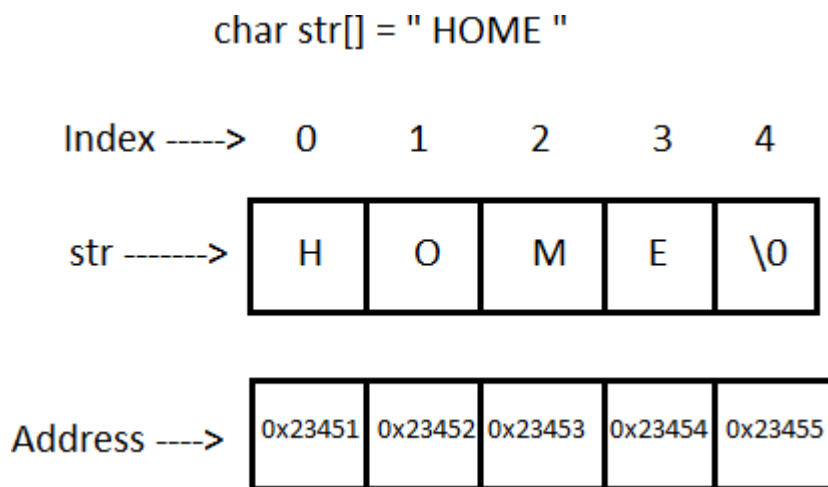
Title: Write a Java/C/C++/Python program that contains a string (char pointer) with a value 'Hello World'. The program should AND or and XOR each character in this string with 127 and display the result.

String: string is a data type used in programming, such as an integer and floating point unit, but is used to represent text rather than numbers. a string is a **sequence of characters terminated with a null character \0** . For example: `char c[] = "c string"`

The string declaration can be done in two ways.

1) Declaration of string by char array in c lang:

For example: The string "home" contains 5 characters including the '\0' character which is automatically added by the compiler at the end of the string.



2) Declaration of string by string by the string literal in c lang:

A "string literal" is a sequence of characters from the source character set enclosed in double quotation marks (" "). String literals are used to represent a sequence of characters which, taken together, form a null-terminated string.

For example:

`Char str()="thisismybooks";`

In such case, '\0' will be appended at the end of the string by the compiler.

(b) AND operation:

This operation is also a very important in logical operation in [digital electronics](#). In other words **Logical AND operation** is also known as **Boolean AND operation**. **AND operation** is different from OR operation. Like Logical OR operation this is also very useful in digital circuits and arithmetical solves. There are several I.Cs where the concept of **Logical AND operation** is used. Again to understand these types of operation we have to go through the [truth table](#). In **Boolean AND operation** the sign used is (.). Here also we have two inputs and a single output. Now let us follow the truth table of **AND operation**.



Logical AND Gate

Fig - 2

Here the outputs are quite different from OR operation. Traverse the table row by row. Firstly the inputs are $A = 0$ and $B = 0$ for which after logical AND operation the output is $Y = 0$. Again traverse to the next row, here the inputs are $A = 0$ and $B = 1$. The respective output is 0. Now the inputs are $A = 1$ and $B = 0$ and the output after AND operation is also 0.

| INPUT | | OUTPUT |
|-------|---|-----------|
| A | B | $Y = A.B$ |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

(c) XOR operation:

The XOR logical operation, exclusive or, **takes two boolean operands and returns true if, and only if, the operands are different**. Conversely, it returns false if the two operands have the same value.

| XOR truth table | | |
|-----------------|---|--------|
| Input | | Output |
| A | B | |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

- 0, false
- 1, true

Algorithm:

1. Start
2. Take the input 'hello world' which is assigned to variable 'str'.
3. Perform AND and XOR operation between the string and 127.
4. Print the result.
5. Stop.

Source code:

```
string = "hello
world" a =
len(string)
for x in range(a):

    ans1=x & 127

    print("AND OPERATION",ans1)

for y in range(a):

    ans2=y | 127

    print("OR OPERATION]",
ans2)for z in range(a):
    ans3=2 ^ 127

    print("EXOR OPERATION", ans3)
```

Output:

Programiz
Python Online Compiler

Learn Python App

main.py

Run

Shell

Clear

```
1 string = "hello world"
2 a = len(string)
3 for x in range(a):
4
5     ans1=x & 127
6
7 print("AND OPERATION",ans1)
8
9 for y in range(a):
10
11     ans2=y | 127
12
13 print("OR OPERATION]", ans2)
14 for z in range(a):
15     ans3=2 ^ 127
16
17 print("EXOR OPERATION", ans3)
18
```

```
AND OPERATION 10
OR OPERATION] 127
EXOR OPERATION 125
>
```

PRACTICAL NO:8

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

Theory:

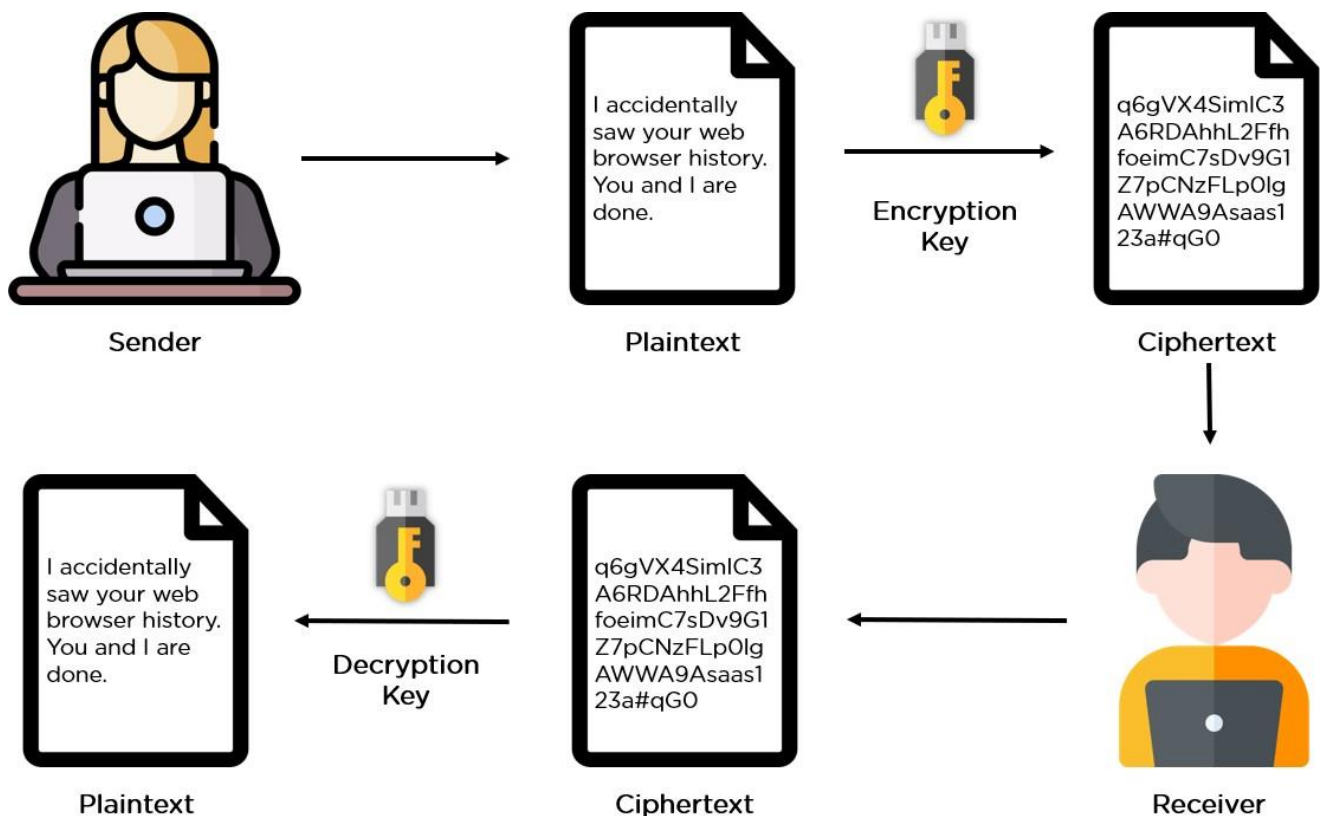
Encryption:

Data encryption is the process of converting data from a readable format to a scrambled piece of information. This is done to prevent prying eyes from reading confidential data in transit. [Encryption](#) can be applied to documents, files, messages, or any other form of communication over a network.

Example

A woman wants to send her boyfriend a personal text, so she encrypts it using specialized software that scrambles the data into what appears to be unreadable gibberish. She then sends the message out, and her boyfriend, in turn, uses the correct decryption to translate it.

Thus, what starts out looking like this:

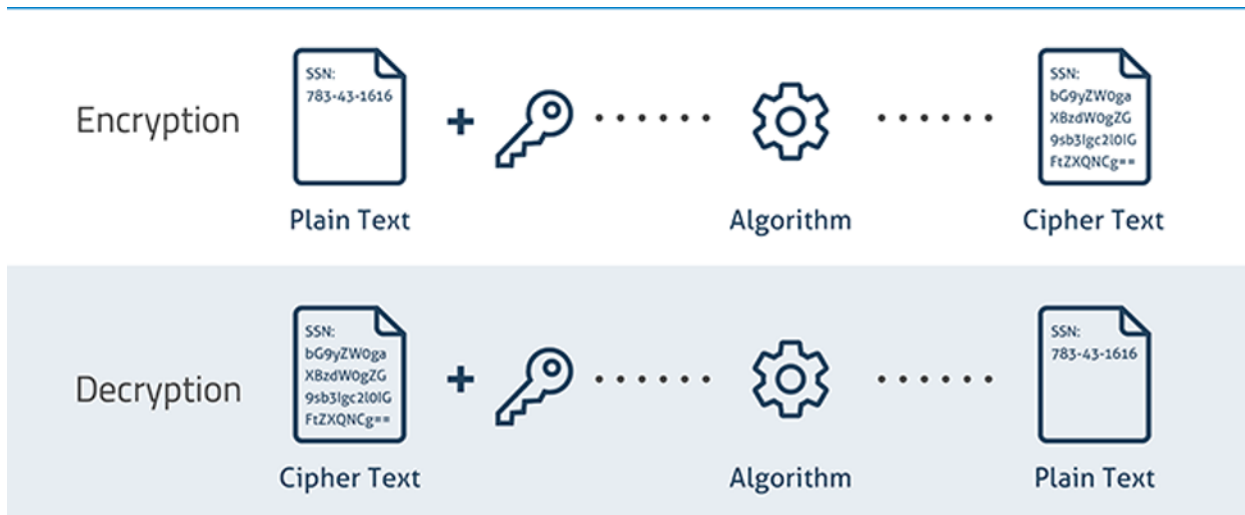


Fortunately, the keys do all the actual encryption/decryption work, leaving both people more time to contemplate the smoldering ruins of their relationship in total privacy.

Decryption: The conversion of encrypted data into its original form is called Decryption. It is generally a reverse process of encryption. It decodes the encrypted information so that an authorized user can only decrypt the data because decryption requires a secret key or password.

Example:

“Absent the tools for decryption, someone who intercepts our message cannot access our documents in any readable format.”



Transposition cipher: In cryptography, a **transposition cipher** is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed (the plaintext is reordered). Mathematically function is used on the characters' positions to encrypt and an inverse function to decrypt.

Example:

A simple example for a transposition cipher is **columnar transposition cipher** where each character in the plain text is written horizontally with specified alphabet width. The cipher is written vertically, which creates an entirely different cipher text.

Consider the plain text **hello world**, and let us apply the simple columnar transposition technique as shown below

| | | | |
|---|---|---|---|
| h | e | l | l |
| o | w | o | r |
| l | d | | |

The plain text characters are placed horizontally and the cipher text is created with vertical format as : **holewdlo lr**. Now, the receiver has to use the same table to decrypt the cipher text to plain text.

Transposition Techniques:

Transposition Techniques are based on the permutation of the plain-text instead of substitution.

1) Rail-Fence Technique

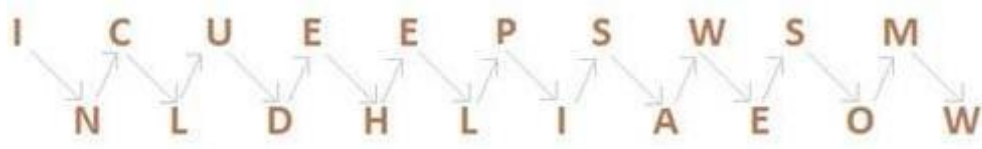
This technique is a type of Transposition technique and does is write the plain text as a sequence of diagonals and changing the order according to each row.

It uses a simple algorithm,

1. Writing down the plaintext message into a sequence of diagonals.
2. Row-wise writing the plain-text written from above step.

Example,

Let's say, we take an example of "INCLUDEHELP IS AWESOME".



So the Cipher-text are, **ICUEEPSWSMNL DHLIAEOW**.

First, we write the message in a zigzag manner then read it out direct row-wise to change it to cipher-text.

Now as we can see, Rail-Fence Technique is very to break by any cryptanalyst.

2) Columnar Transition Technique

A. Basic Technique

It is a slight variation to the Rail-fence technique, let's see its algorithm:

1. In a rectangle of pre-defined size, write the plain-text message row by row.
2. Read the plain message in random order in a column-wise fashion. It can be any order such as 2, 1, 3 etc.
3. Thus Cipher-text is obtained.

Let's see an example:

Original message: **"INCLUDEHELP IS AWESOME"**.

Now we apply the above algorithm and create the rectangle of 4 columns (we decide to make a rectangle with four columns it can be any number.)

| Column 1 | Column 2 | Column 3 | Column 4 |
|----------|----------|----------|----------|
| I | N | C | L |
| U | D | E | H |
| E | L | P | I |
| S | A | W | E |
| S | O | M | E |

Now let's decide on an order for the column as 4, 1, 3 and 2 and now we will read the text in column-wise.

Cipher-text: **LHIEEIUESSCEPWMNDLAO**

B. Columnar Technique with multiple rounds

In this method, we again change the cipher text we received from a Basic technique that is in round 1 and again follows the same procedure for the cipher-text from round 1.

Algorithm:

1. In a rectangle of pre-defined size, write the plain-text message row by row.
2. Read the plain message in random order in a column-wise fashion. It can be any order such as 2, 1, 3 etc.
3. Thus, Cipher-text of round 1 is obtained.
4. Repeat from step 1 to 3.

Example:

Original message: "INCLUDEHELP IS AWESOME".

Now we apply the above algorithm and create the rectangle of 4 column (we decide to make a rectangle with four column it can be any number.)

| Column 1 | Column 2 | Column 3 | Column 4 |
|----------|----------|----------|----------|
| I | N | C | L |
| U | D | E | H |
| E | L | P | I |
| S | A | W | E |
| S | O | M | E |

Now let's decide on an order for the column as 4, 1, 3 and 2 and now we will read the text in column-wise.

Cipher-text of round 1: **LHIEEIUESSCEPWMNDLAO**

Round 2:

| Column 1 | Column 2 | Column 3 | Column 4 |
|----------|----------|----------|----------|
| L | H | I | E |
| E | I | U | E |
| S | S | C | E |
| P | W | M | N |
| D | L | A | O |

Now, we decide to go with a previous order that is 4,1,3,2.

Cipher-text: **EEENLESPICUMHISW**

These multi-round columnar techniques are harder to crack as compared to methods seen earlier.

Algorithm:

Columnar transposition techniques

Step 1: Write all character of plain text message row by row in a rectangle of predefined size.

Step 2: Read the message in a columnar maner, i.e. column by column.

Step3: The resultant message is ciphertext.

Source code:

```
Import math
```

```
Key=input("enter the key")
```

```
# Encryption
```

```
Def encryptMessage(msg):
```

```
    Cipher = ""
```

```
    K_indx = 0
```

```
    Msg_len = float(len(msg))
```

```
    Msg_lst = list(msg)
```

```
    Key_lst = sorted(list(key))
```

```
    Col = len(key)
```

```
    Row = int(math.ceil(msg_len / col))
```

```
    Fill_null = int((row * col) - msg_len)
```

```
    Msg_lst.extend('_' * fill_null)
```

```
Matrix = [msg_lst[i: I + col]
```

```
For I in range(0, len(msg_lst), col)]
```

```
For _ in range(col):
```

```
    Curr_idx = key.index(key_lst[k_indx])
```

```
    Cipher += ''.join([row[curr_idx]
```

```
                        For row in matrix])
```

```
    K_indx += 1
```

```
Return cipher
```

```
# Decryption
```

```
Def decryptMessage(cipher):
```

```
    Msg = ""
```

```
    K_indx = 0
```



```
Msg_idx = 0
```

```
Msg_len = float(len(cipher))
```

```
Msg_lst = list(cipher)
```

```
Col = len(key)
```

```
Row = int(math.ceil(msg_len / col))
```

```
Key_lst = sorted(list(key))
```

```
Dec_cipher = []
```

```
For _ in range(row):
```

```
    Dec_cipher += [[None] * col]
```

```
For _ in range(col):
```

```
    Curr_idx = key.index(key_lst[k_idx])
```

```
    For j in range(row):
```

```
        Dec_cipher[j][curr_idx] = msg_lst[msg_idx]
```

```
        Msg_idx += 1
```

```
    K_idx += 1
```

Try:

```
Msg = ''.join(sum(dec_cipher, []))
```

Except TypeError:

```
Raise TypeError("This program cannot",
```

```
                "handle repeating words.")
```

```
Null_count = msg.count('_')
```

If null_count > 0:

```
Return msg[: -null_count]
```

Return msg

```
Msg = input("enter the msg")
```

```
Cipher = encryptMessage(msg)
```

```
Print("Encrypted Message: {}".
```

```
Format(cipher))
```

```
Print("Decryped Message: {}".
```

```
Format(decryptMessage(cipher)))
```

Output:

The screenshot displays the OnlineGDB IDE interface. The main editor shows a Python script for a Vigenere cipher. The code includes functions for encryption and decryption, and a main loop that takes user input for a key and a message. The output window shows the following sequence of events:

```
enter the keyHELLO
enter the msgPUNE
Encrypted Message: UPNN
Traceback (most recent call last):
  File "main.py", line 53, in decryptMessage
    msg = ''.join(sum(dec_cipher, []))
TypeError: sequence item 3: expected str instance, NoneType found
```

The error message indicates a `TypeError` in the `decryptMessage` function, specifically in the line where the decrypted message is constructed using `sum(dec_cipher, [])`. The error suggests that the `dec_cipher` variable contains a `NoneType` instance, which is not compatible with the `sum` function's requirements.

PRACTICAL NO 9

Title: Write a Java/C/C++/Python program to implement DES algorithm.

Theory:

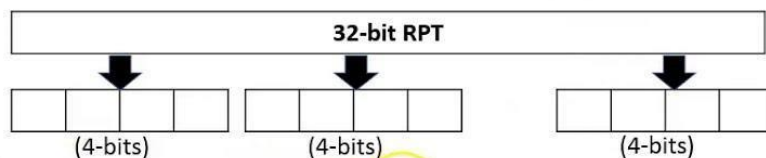
DES Algorithm: The DES (Data Encryption Standard) algorithm is a symmetric-key block cipher created in the early 1970s by an IBM team and adopted by the National Institute of Standards and Technology (NIST). The algorithm takes the plain text in 64-bit blocks and converts them into ciphertext using 48-bit 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.

Algorithm:

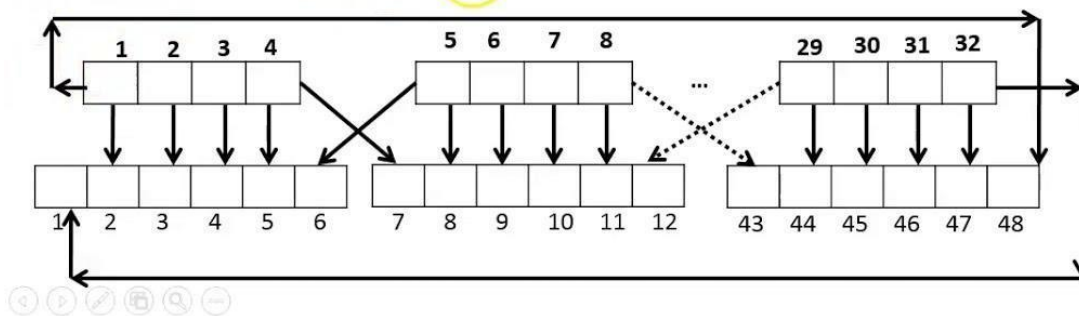
DES (Data Encryption Standard)

Expansion Permutation

- 32-bit RPT of IP is expanded to 48-bits
- Expansion permutation steps:
 - 32-bit RPT is divided into 8-blocks each of 4-bits



- Each 4-bit block is expanded to 6-bit and produce 48-bit output



The algorithm process breaks down into the following steps:

1. The process begins with the 64-bit plain text block getting handed over to an initial permutation (IP) function.
2. The initial permutation (IP) is then performed on the plain text.
3. Next, the initial permutation (IP) creates two halves of the permuted block, referred to as Left Plain Text (LPT) and Right Plain Text (RPT).

4. Each LPT and RPT goes through 16 rounds of the encryption process.
5. Finally, the LPT and RPT are rejoined, and a Final Permutation (FP) is performed on the newly combined block.
6. The result of this process produces the desired 64-bit cipher text.

Source code:

```
import base32hex
import hashlib
from Crypto.Cipher import
DESpassword = "Password"
salt = '\x28\xAB\xBC\xCD\xDE\xEF\x00\x33'
key = password + salt
m =
hashlib.md5(key)key
= m.digest()
(dk, iv)=(key[:8], key[8:])
crypter = DES.new(dk, DES.MODE_CBC, iv)

plain_text= "I see you"

print("The plain text is : ",plain_text)
plain_text += '\x00' * (8 - len(plain_text) %
8)ciphertext = crypter.encrypt(plain_text)
encode_string= base32hex.b32encode(ciphertext)
print("The encoded string is : ",encode_string)
```

Output After Encryption:

```
D:\Scripts\UEM>py -2.7 encrypt_foo.py
('The plain text is : ', 'I see you')
('The encoded string is : ', 'UH562EGM8RCHHTOUC5CTRS59OG=====')

D:\Scripts\UEM>
```

The Code For Decryption Is Mentioned Below:

```
import base32hex
import hashlib
from Crypto.Cipher import DES
```

```
password = "Password"
salt = '\x28\xAB\xBC\xCD\xDE\xEF\x00\x33'
key = password + salt
m =
hashlib.md5(key).digest()
(dk, iv) = (key[:8], key[8:])
crypter = DES.new(dk, DES.MODE_CBC, iv)

encrypted_string = 'UH562EGM8RCHHTOUC5CTRS59OG====='

print("The encrypted string is : ", encrypted_string)
encrypted_string = base32hex.b32decode(encrypted_string)
decrypted_string = crypter.decrypt(encrypted_string)
print("The decrypted string is : ", decrypted_string)
```

Output After Decryption

```
D:\Scripts\UEM>py -2.7 e.py
The encrypted string is : UH562EGM8RCHHTOUC5CTRS59OG====
The plain text is : I see you

D:\Scripts\UEM>
```

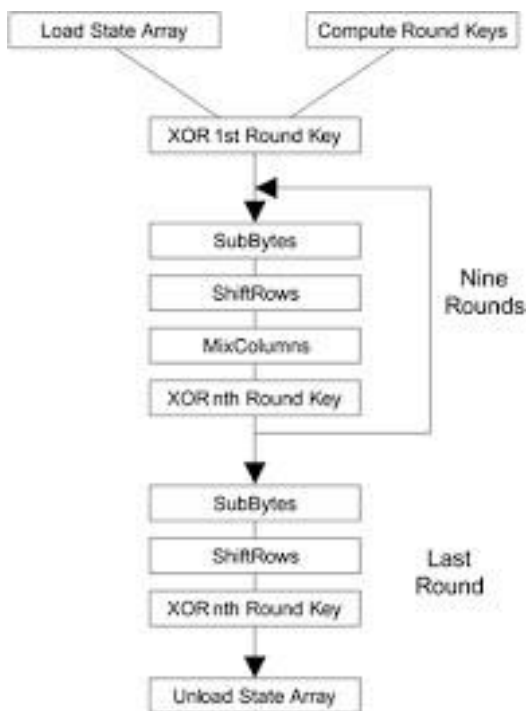
PRACTICAL NO 10

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.

The algorithm process breaks down into the following steps:



1. Derive the set of round keys from the cipher key.
2. Initialize the state array with the block data (plaintext).
3. Add the initial round key to the starting state array.
4. Perform nine rounds of state manipulation.
5. Perform the tenth and final round of state manipulation.
6. Copy the final state array out as the encrypted data (cipher text).

Source code:

```
import
java.security.MessageDigest;
import java.util.Arrays;
import
javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
```

```

import javax.crypto.spec.IvParameterSpec;

import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class AES {
    static String IV = "AAAAAAAAAAAAAAAAAAAA";
    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");
}
}

```

Output:

The screenshot shows the Programiz Java Online Compiler interface. The editor contains a Java program that demonstrates AES encryption and decryption. The program defines a static IV, a plaintext string, and an encryption key. It then performs encryption and decryption, printing the results to the console.

```

1- import java.security.MessageDigest;
2- import java.util.Arrays;
3- import javax.crypto.KeyGenerator;
4- import javax.crypto.SecretKey;
5- import javax.crypto.spec.SecretKeySpec;
6- import javax.crypto.spec.IvParameterSpec;
7-
8- import javax.crypto.Cipher;
9- import javax.crypto.spec.IvParameterSpec;
10- import javax.crypto.spec.SecretKeySpec;
11-
12- public class AES {
13-     static String IV = "AAAAAAAAAAAAAAAA";
14-     static String plaintext = "test text 123\u0000\u0000\u0000\u0000"; /*Note null
padding*/
15-     static String encryptionKey = "0123456789abcdef";
16-     public static void main(String [] args) {
17-         try {
18-
19-             System.out.println("==Java==");
20-             System.out.println("plain:  " + plaintext);
21-
22-             byte[] cipher = encrypt(plaintext, encryptionKey);
23-
24-             System.out.print("cipher:  ");

```

The output window shows the following results:

```

java -cp /tmp/OBNK15q9Io AES
==Java==
plain:  test text 123...
cipher:  16 -124 41 -83 -16 -123 61 -64 -15 -74 87 28 63 30 64 78
decrypt: test text 123...

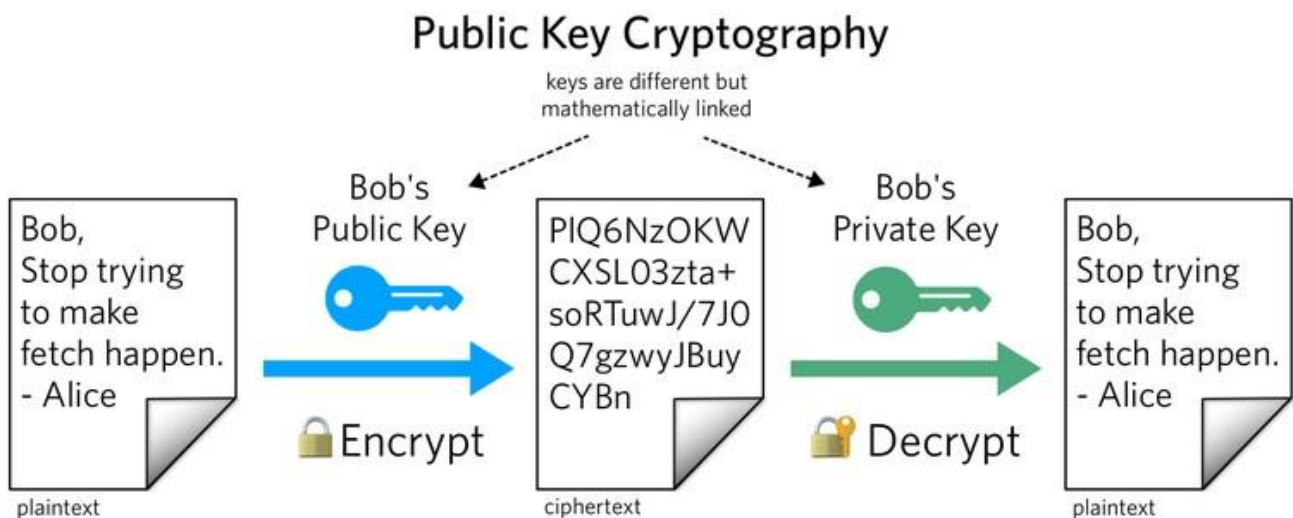
```

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 the oldest. 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 given to everyone and Private key is kept private.



An example of asymmetric cryptography:

1. A client (for example browser) sends its public key to the server and requests for some data.
2. The server encrypts the data using client's public key and sends the encrypted data.
3. Client receives this data and decrypts it.

Algorithm:

The algorithm process breaks down into the following steps:

Step-1: Choose two prime number and. Lets take and.

Step-2: Compute the value of and. It is given as, and.

Step-3: Find the value of (public key) Choose, such that should be co-prime.

Step-4: Compute the value of (private key).

Step-5: Do the encryption and decryption.

Source code:

// Java Program to Implement the RSA Algorithm

import java.math.*;

import java.util.*;

class RSA {

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);
}
}

```

Output:

Programiz

Java Online Compiler

Learn Java App

Main.java

Run

Clear

```
1 // Java Program to Implement the RSA Algorithm
2 import java.math.*;
3 import java.util.*;
4
5 class RSA {
6     public static void main(String args[])
7     {
8         int p, q, n, z, d = 0, e, i;
9
10        // The number to be encrypted and decrypted
11        int msg = 12;
12        double c;
13        BigInteger msgback;
14
15        // 1st prime number p
16        p = 3;
17
18        // 2nd prime number q
19        q = 11;
20        n = p * q;
21        z = (p - 1) * (q - 1);
22        System.out.println("the value of z = " + z);
23
24        for (e = 2; e < z; e++) {
25
```

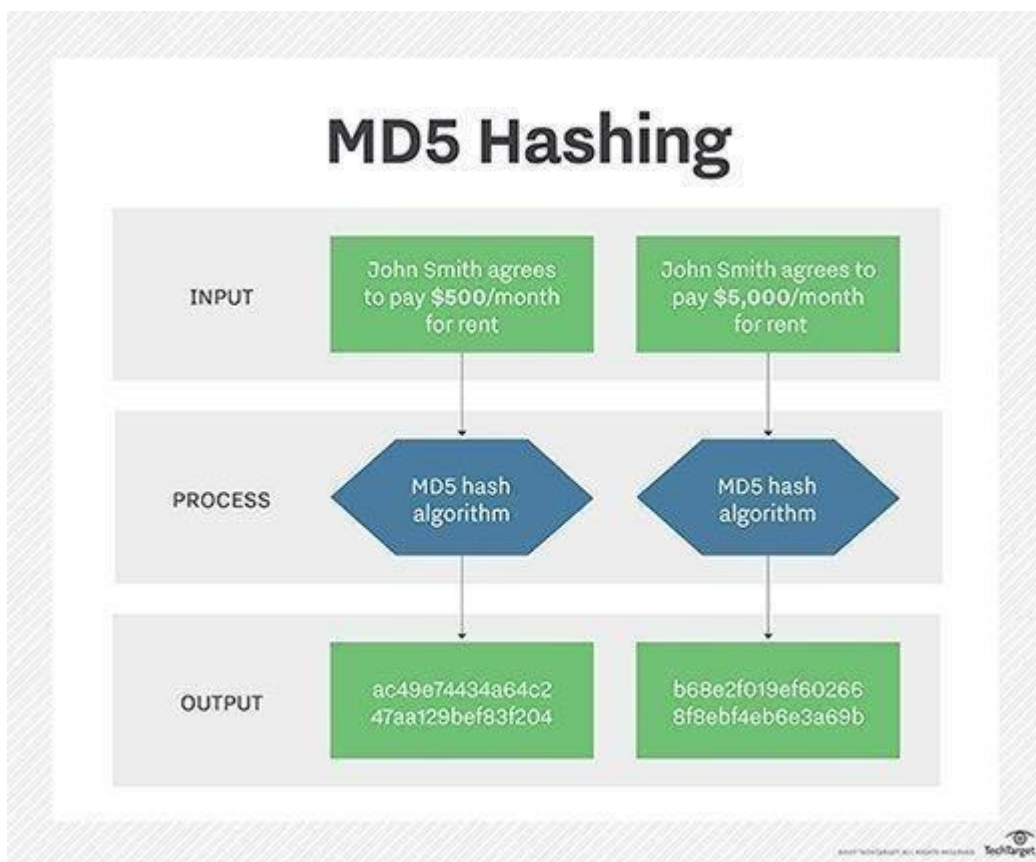
```
java -cp /tmp/73xjOQYqvF RSA
the value of z = 20the value of e = 3the value of d = 7Encrypted message is : 12
.0Decrypted message is : 12
```

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.



MD5 mainly used for securing password in database server.

Algorithm:

The algorithm process breaks down into the following steps:

Step1: Append Padding Bits. Padding means adding extra bits to the original message.

Step 2: Append Length. After padding, 64 bits are inserted at the end, which is used to record the original input length.

Step 3: Initialize MD buffer.

Step 4: Processing message in 16-word block.

Source code:

```
import java.nio.charset.Charset;

import java.nio.charset.StandardCharsets;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;


public class MD5Utils {


    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 = MD5Utils.digest(pText.getBytes(UTF_8));

    System.out.println(String.format(OUTPUT_FORMAT, "MD5 (hex) ", bytesToHex(md5InBytes)));

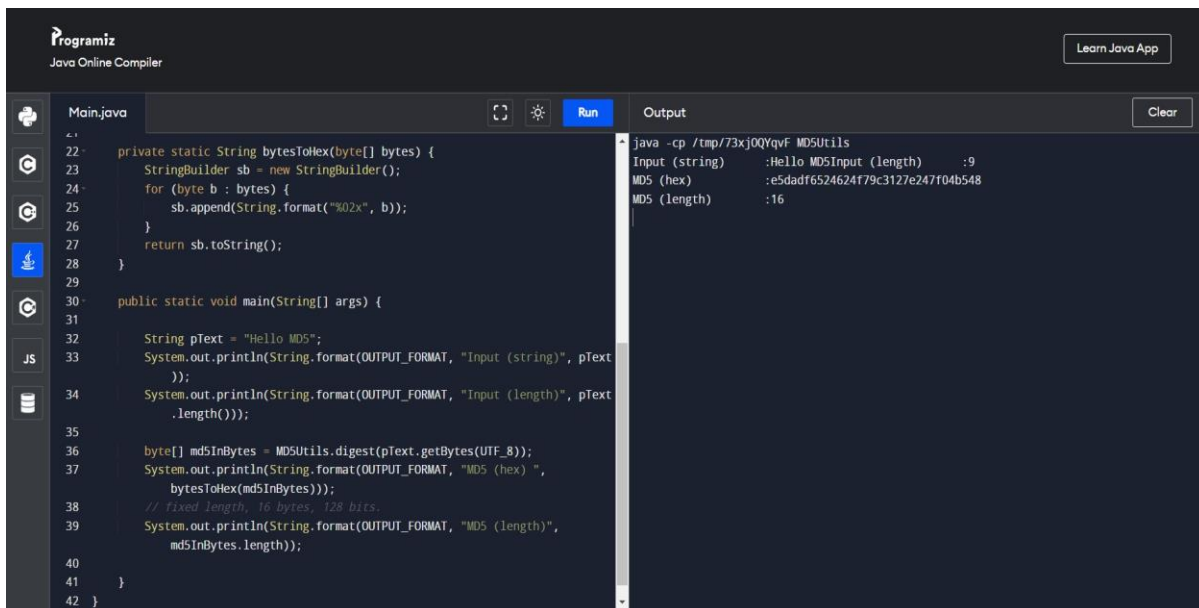
    // fixed length, 16 bytes, 128 bits.

    System.out.println(String.format(OUTPUT_FORMAT, "MD5 (length)", md5InBytes.length));

}
}

```

Output:



The screenshot shows the Programiz Java Online Compiler interface. The left pane displays the source code of a Java program. The right pane shows the output of the program after clicking the 'Run' button.

Source Code (Main.java):

```

22- private static String bytesToHex(byte[] bytes) {
23-     StringBuilder sb = new StringBuilder();
24-     for (byte b : bytes) {
25-         sb.append(String.format("%02x", b));
26-     }
27-     return sb.toString();
28- }
29-
30- public static void main(String[] args) {
31-
32-     String pText = "Hello MD5";
33-     System.out.println(String.format(OUTPUT_FORMAT, "Input (string)", pText
34- ));
35-     System.out.println(String.format(OUTPUT_FORMAT, "Input (length)", pText
36- .length()));
37-
38-     byte[] md5InBytes = MD5Utils.digest(pText.getBytes(UTF_8));
39-     System.out.println(String.format(OUTPUT_FORMAT, "MD5 (hex) ",
40- bytesToHex(md5InBytes)));
41-     // fixed length, 16 bytes, 128 bits.
42-     System.out.println(String.format(OUTPUT_FORMAT, "MD5 (length)",
43- md5InBytes.length));
44- }
45- }

```

Output:

```

java -cp /tmp/73xj0QYqvF MD5Utils
Input (string)      :Hello MD5Input (length)      :9
MD5 (hex)          :e5dadf6524624f79c3127e247f04b548
MD5 (length)       :16

```