Day -3

## Assignment

## Menda Mani Sai

## 192111399

### 1. Write a program to find the reverse of a given number.

#include <stdio.h>

int main() {

int num, reversed = 0;

printf("Enter a number: ");

scanf("%d", &num);

while (num != 0) {

int remainder = num % 10;

reversed = reversed \* 10 + remainder;

num = num / 10;

}

printf("The reverse of the number is: %d\n", reversed);

return 0;

}

#### OUTPUT:

### 2. Write a program to find the perfect number.

#include <stdio.h>

int isPerfect(int num) {

int sum = 0;

for (int i = 1; i <= num / 2; i++) {

if (num % i == 0) {

sum += i;

}

}

return sum == num;

}

int main() {

int n;

printf("Enter a positive integer: ");

scanf("%d", &n);

if (isPerfect(n)) {

printf("%d is a perfect number.\n", n);

} else {

printf("%d is not a perfect number.\n", n);

}

return 0;

}

#### OUTPUT:

### 3. Write a program to perform travelling salesman problem using dynamic programming

#include <stdio.h>

#include <limits.h>

#define MAXCITIES 20

int n;

int dist[MAXCITIES][MAXCITIES];

int memo[MAXCITIES][1 << MAXCITIES];

int VISITED\_ALL;

int tsp(int current, int mask) {

if (mask == VISITED\_ALL) {

return dist[current][0];

}

if (memo[current][mask] != -1) {

return memo[current][mask];

}

int minCost = INT\_MAX;

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

if ((mask & (1 << city)) == 0) {

int newCost = dist[current][city] + tsp(city, mask | (1 << city));

if (newCost < minCost) {

minCost = newCost;

}

}

}

memo[current][mask] = minCost;

return minCost;

}

int main() {

printf("Enter the number of cities: ");

scanf("%d", &n);

printf("Enter the distance matrix:\n");

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

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

scanf("%d", &dist[i][j]);

}

}

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

for (int j = 0; j < (1 << n); j++) {

memo[i][j] = -1;

}

}

VISITED\_ALL = (1 << n) - 1;

int minCost = tsp(0, 1);

printf("Minimum cost for TSP: %d\n", minCost);

return 0;

}

#### OUTPUT:

### 4. Write a program for the given pattern

### If n=4 1

### 1 2

### 1 2 3

### 1 2 3 4

#include<stdio.h>

int main()

{

int a,i,j;

printf("Enter the value of n= ");

scanf("%d",&a);

for(i=1;i<=a;i++){

for(j=1;j<=i;j++){

printf("%d",j);

}

printf("\n");

}

}

#### OUTPUT:

### 5. Write a program to perform Floyd’s algorithm

#include <stdio.h>

#include <limits.h>

#define V 100

void floydWarshall(int graph[V][V], int n) {

int dist[V][V];

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

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

dist[i][j] = graph[i][j];

}

}

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

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

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

if (dist[i][k] != INT\_MAX && dist[k][j] != INT\_MAX &&

dist[i][k] + dist[k][j] < dist[i][j]) {

dist[i][j] = dist[i][k] + dist[k][j];

}

}

}

}

printf("Shortest distances between all pairs of vertices:\n");

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

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

if (dist[i][j] == INT\_MAX) {

printf("INF\t");

} else {

printf("%d\t", dist[i][j]);

}

}

printf("\n");

}

}

int main() {

int n;

printf("Enter the number of vertices: ");

scanf("%d", &n);

int graph[V][V];

printf("Enter the adjacency matrix with distances (use 'INF' for infinity):\n");

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

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

scanf("%d", &graph[i][j]);

if (graph[i][j] == -1) {

graph[i][j] = INT\_MAX;

}

}

}

floydWarshall(graph, n);

return 0;

}

#### OUTPUT:

### 6. Write a program for pascal triangle.

#include <stdio.h>

int binomialCoefficient(int n, int k) {

if (k == 0 || k == n)

return 1;

return binomialCoefficient(n - 1, k - 1) + binomialCoefficient(n - 1, k);

}

void printPascalsTriangle(int numRows) {

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

for (int j = 0; j < numRows - i; j++) {

printf(" ");

}

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

int coef = binomialCoefficient(i, j);

printf("%d ", coef);

}

printf("\n");

}

}

int main() {

int numRows;

printf("Enter the number of rows for Pascal's Triangle: ");

scanf("%d", &numRows);

if (numRows <= 0) {

printf("Please enter a positive number of rows.\n");

} else {

printf("Pascal's Triangle with %d rows:\n", numRows);

printPascalsTriangle(numRows);

}

return 0;

}

#### OUTPUT:

### 7. Write a program to find the optimal cost by using appropriate algorithm

#include <stdio.h>

#define MAX\_ITEMS 100

#define MAX\_WEIGHT 100

int weights[MAX\_ITEMS];

int values[MAX\_ITEMS];

int numItems;

int maxWeight;

int max(int a, int b) {

return (a > b) ? a : b;

}

int knapsack(int item, int weight) {

if (item == numItems || weight == 0) {

return 0;

}

if (weights[item] > weight) {

return knapsack(item + 1, weight);

} else {

int include = values[item] + knapsack(item + 1, weight - weights[item]);

int exclude = knapsack(item + 1, weight);

return max(include, exclude);

}

}

int main() {

printf("Enter the number of items: ");

scanf("%d", &numItems);

printf("Enter the maximum weight: ");

scanf("%d", &maxWeight);

printf("Enter the weight and value of each item:\n");

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

scanf("%d %d", &weights[i], &values[i]);

}

int optimalValue = knapsack(0, maxWeight);

printf("The optimal value is: %d\n", optimalValue);

return 0;

}

#### OUTPUT:

### 8. Write a program to find the sum of digits.

#include <stdio.h>

int main() {

int num, sum = 0;

printf("Enter an integer: ");

scanf("%d", &num);

while (num != 0) {

int digit = num % 10;

sum += digit;

num /= 10;

}

printf("Sum of digits: %d\n", sum);

return 0;

}

#### OUTPUT:

### 9. Write a program to print minimum and maximum value sequency for all the numbers in a

### list.

#include <stdio.h>

#include <limits.h>

int main() {

int n;

printf("Enter the number of elements in the list: ");

scanf("%d", &n);

int list[n];

printf("Enter the elements of the list:\n");

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

scanf("%d", &list[i]);

}

printf("Minimum and Maximum Value Subsequences:\n");

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

int minVal = INT\_MAX;

int maxVal = INT\_MIN;

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

if (list[j] < minVal) {

minVal = list[j];

}

if (list[j] > maxVal) {

maxVal = list[j];

}

printf("For number %d: Min: %d, Max: %d\n", list[i], minVal, maxVal);

}

}

return 0;

}

#### OUTPUT:

### 10. Write a program to perform n Queens Problem using backtracking.

#include <stdio.h>

#include <stdbool.h>

bool isSafe(int board[][100], int row, int col, int n) {

if (board[i][col] == 1) {

return false;

}

}

for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {

if (board[i][j] == 1) {

return false;

}

}

for (int i = row, j = col; i >= 0 && j < n; i--, j++) {

if (board[i][j] == 1) {

return false;

}

}

return true;

}

void printBoard(int board[][100], int n) {

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

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

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

}

printf("\n");

}

}

bool solveNQueens(int board[][100], int row, int n) {

if (row == n) {

printBoard(board, n);

return true;

}

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

if (isSafe(board, row, col, n)) {

board[row][col] = 1;

if (solveNQueens(board, row + 1, n)) {

return true;

}

board[row][col] = 0;

}

}

return false;

}

int main() {

int n;

printf("Enter the board size (N): ");

scanf("%d", &n);

int board[100][100] = {0};

if (!solveNQueens(board, 0, n)) {

printf("No solution exists for N=%d.\n", n);

}

return 0;

}

OUTPUT: