**Code**

#*include* <stdio.h> #*include* <stdbool.h>

#*define N* 5

int knightMoves[8][2] = {

{2, 1}, {1, 2}, {-1, 2}, {-2, 1},

{-2, -1}, {-1, -2}, {1, -2}, {2, -1}

};

*bool isSafe*(int x, int y, int board[*N*][*N*]) {

*return* (x >= 0 && x < *N* && y >= 0 && y < *N* && board[x][y] == -1);

}

*bool knightTour*(int x, int y, int movei, int board[*N*][*N*]) {

*if* (movei == *N* \* *N*) {

*return true*;

}

*for* (int i = 0; i < 8; i++) {

int newX = x + knightMoves[i][0]; int newY = y + knightMoves[i][1];

*if* (*isSafe*(newX, newY, board)) { board[newX][newY] = movei;

*if* (*knightTour*(newX, newY, movei + 1, board)) {

*return true*;

}

board[newX][newY] = -1;

}

}

*return false*;

}

void *printBoard*(int board[*N*][*N*]) {

*for* (int i = 0; i < *N*; i++) {

*for* (int j = 0; j < *N*; j++) {

*printf*("%2d ", board[i][j]);

}

*printf*("\n");

}

*printf*("\n");

}

*bool solveKnightTour*() { int board[*N*][*N*];

*for* (int i = 0; i < *N*; i++) {

*for* (int j = 0; j < *N*; j++) { board[i][j] = -1;

}

}

board[0][0] = 0;

*if* (*knightTour*(0, 0, 1, board)) {

*printBoard*(board);

*return true*;

}

*return false*;

}

int *main*() {

*if* (!*solveKnightTour*()) {

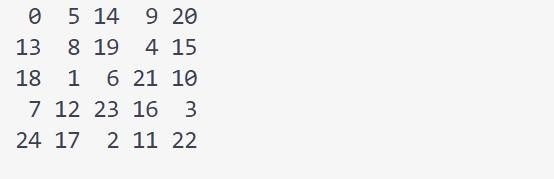
*printf*("No solution exists!\n");

}

*return* 0;

}

**Output**



**Code**

#*include* <stdio.h> #*include* <string.h>

void *computeLPSArray*(char \*pattern, int m, int \*lps) { int len = 0;

int i = 1;

lps[0] = 0;

*while* (i < m) {

*if* (pattern[i] == pattern[len]) { len++;

lps[i] = len; i++;

} *else* {

*if* (len != 0) {

len = lps[len - 1];

} *else* {

lps[i] = 0; i++;

}

}

}

}

void *KMPSearch*(char \*text, char \*pattern) { int n = *strlen*(text);

int m = *strlen*(pattern); int lps[m];

*computeLPSArray*(pattern, m, lps);

int i = 0, j = 0;

*while* (i < n) {

*if* (pattern[j] == text[i]) { i++;

j++;

}

*if* (j == m) {

*printf*("Pattern found at index %d\n", i - j); j = lps[j - 1];

} *else if* (i < n && pattern[j] != text[i]) {

*if* (j != 0)

j = lps[j - 1];

*else*

i++;

}

}

}

int *main*() {

char text*[]* = "ABABDABACDABABCABAB";

char pattern*[]* = "ABABCABAB"; *KMPSearch*(text, pattern); *return* 0;

}

**Output**



**Code**

#*include* <stdio.h>

#*define MAX* 4

void *addMatrix*(int A[*MAX*][*MAX*], int B[*MAX*][*MAX*], int C[*MAX*][*MAX*], int n) {

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

*for* (int j = 0; j < n; j++) C[i][j] = A[i][j] + B[i][j];

}

void *subtractMatrix*(int A[*MAX*][*MAX*], int B[*MAX*][*MAX*], int C[*MAX*][*MAX*], int n) {

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

*for* (int j = 0; j < n; j++) C[i][j] = A[i][j] - B[i][j];

}

void *strassen*(int A[*MAX*][*MAX*], int B[*MAX*][*MAX*], int C[*MAX*][*MAX*], int n) {

*if* (n == 1) {

C[0][0] = A[0][0] \* B[0][0];

*return*;

}

int mid = n / 2;

int A11[*MAX*][*MAX*], A12[*MAX*][*MAX*], A21[*MAX*][*MAX*], A22[*MAX*][*MAX*];

int B11[*MAX*][*MAX*], B12[*MAX*][*MAX*], B21[*MAX*][*MAX*], B22[*MAX*][*MAX*];

int M1[*MAX*][*MAX*], M2[*MAX*][*MAX*], M3[*MAX*][*MAX*], M4[*MAX*][*MAX*];

int M5[*MAX*][*MAX*], M6[*MAX*][*MAX*], M7[*MAX*][*MAX*];

int temp1[*MAX*][*MAX*], temp2[*MAX*][*MAX*];

*for* (int i = 0; i < mid; i++) {

*for* (int j = 0; j < mid; j++) { A11[i][j] = A[i][j];

A12[i][j] = A[i][j + mid];

A21[i][j] = A[i + mid][j];

A22[i][j] = A[i + mid][j + mid];

B11[i][j] = B[i][j];

B12[i][j] = B[i][j + mid];

B21[i][j] = B[i + mid][j];

B22[i][j] = B[i + mid][j + mid];

}

}

*addMatrix*(A11, A22, temp1, mid); *addMatrix*(B11, B22, temp2, mid); *strassen*(temp1, temp2, M1, mid);

*addMatrix*(A21, A22, temp1, mid);

*strassen*(temp1, B11, M2, mid);

*subtractMatrix*(B12, B22, temp2, mid);

*strassen*(A11, temp2, M3, mid);

*subtractMatrix*(B21, B11, temp2, mid);

*strassen*(A22, temp2, M4, mid);

*addMatrix*(A11, A12, temp1, mid);

*strassen*(temp1, B22, M5, mid);

*subtractMatrix*(A21, A11, temp1, mid); *addMatrix*(B11, B12, temp2, mid); *strassen*(temp1, temp2, M6, mid);

*subtractMatrix*(A12, A22, temp1, mid); *addMatrix*(B21, B22, temp2, mid); *strassen*(temp1, temp2, M7, mid);

int C11[*MAX*][*MAX*], C12[*MAX*][*MAX*], C21[*MAX*][*MAX*], C22[*MAX*][*MAX*];

*addMatrix*(M1, M4, temp1, mid); *subtractMatrix*(temp1, M5, temp2, mid); *addMatrix*(temp2, M7, C11, mid);

*addMatrix*(M3, M5, C12, mid);

*addMatrix*(M2, M4, C21, mid);

*addMatrix*(M1, M3, temp1, mid); *subtractMatrix*(temp1, M2, temp2, mid); *addMatrix*(temp2, M6, C22, mid);

*for* (int i = 0; i < mid; i++) {

*for* (int j = 0; j < mid; j++) { C[i][j] = C11[i][j];

C[i][j + mid] = C12[i][j];

C[i + mid][j] = C21[i][j];

C[i + mid][j + mid] = C22[i][j];

}

}

}

int *main*() {

int A[*MAX*][*MAX*], B[*MAX*][*MAX*], C[*MAX*][*MAX*] = {0};

*printf*("Enter elements of matrix A:\n");

*for* (int i = 0; i < *MAX*; i++)

*for* (int j = 0; j < *MAX*; j++)

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

*printf*("Enter elements of matrix B:\n");

*for* (int i = 0; i < *MAX*; i++)

*for* (int j = 0; j < *MAX*; j++)

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

*strassen*(A, B, C, *MAX*);

*printf*("Resultant matrix C:\n");

*for* (int i = 0; i < *MAX*; i++) {

*for* (int j = 0; j < *MAX*; j++) {

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

}

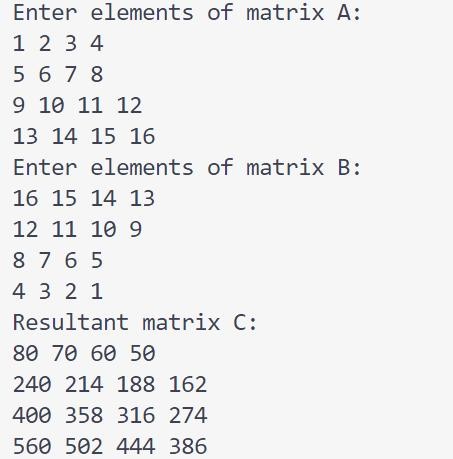
*printf*("\n");

}

*return* 0;

}

**Output**



**Code**

#*include* <stdio.h> #*include* <stdbool.h>

#*define V* 5

*bool isSafe*(int v, int graph[*V*][*V*], int path*[]*, int pos) {

*if* (graph[path[pos - 1]][v] == 0)

*return false*;

*for* (int i = 0; i < pos; i++) {

*if* (path[i] == v)

*return false*;

}

*return true*;

}

*bool hamCycleUtil*(int graph[*V*][*V*], int path*[]*, int pos) {

*if* (pos == *V*) {

*if* (graph[path[pos - 1]][path[0]] == 1)

*return true*; *return false*;

}

*for* (int v = 1; v < *V*; v++) {

*if* (*isSafe*(v, graph, path, pos)) { path[pos] = v;

*if* (*hamCycleUtil*(graph, path, pos + 1))

*return true*; path[pos] = -1;

}

}

*return false*;

}

*bool hamCycle*(int graph[*V*][*V*]) { int path[*V*];

*for* (int i = 0; i < *V*; i++) { path[i] = -1;

}

path[0] = 0;

*if* (*hamCycleUtil*(graph, path, 1) == *false*) { *printf*("Solution does not exist\n"); *return false*;

}

*printf*("Solution Exists: Following is the Hamiltonian Cycle\n");

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

*printf*("%d ", path[i]);

}

*printf*("%d\n", path[0]);

*return true*;

}

int *main*() {

int graph[*V*][*V*] = {

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

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

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

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

{0, 0, 1, 1, 0}

};

*hamCycle*(graph); *return* 0;

}

**Output**

