PROG 1

#include <stdio.h>

void spiralPrint(int m, int n, int arr[m][n]) {

int i, k = 0, l = 0;

while (k < m && l < n) {

for (i = l; i < n; ++i) {

printf("%d ", arr[k][i]);

}

k++;

for (i = k; i < m; ++i) {

printf("%d ", arr[i][n - 1]);

}

n--;

if (k < m) {

for (i = n - 1; i >= l; --i) {

printf("%d ", arr[m - 1][i]);

}

m--;

}

if (l < n) {

for (i = m - 1; i >= k; --i) {

printf("%d ", arr[i][l]);

}

l++;

}}}

int main() {

int arr[3][3] = { {1, 2, 3},

{4, 5, 6},

{7, 8, 9} };

spiralPrint(3, 3, arr);

return 0;

}

PROG 2

#include <stdio.h>

#define N 3

void transpose(int mat[N][N]) {

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

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

int temp = mat[i][j];

mat[i][j] = mat[j][i];

mat[j][i] = temp;

}}}

void reverseRows(int mat[N][N]) {

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

int start = 0, end = N - 1;

while (start < end) {

int temp = mat[i][start];

mat[i][start] = mat[i][end];

mat[i][end] = temp;

start++;

end--;

}}}

void rotate90Clockwise(int mat[N][N]) {

transpose(mat);

reverseRows(mat);

}

void printMatrix(int mat[N][N]) {

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

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

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

}

printf("\n");

}}

int main() {

int mat[N][N] = { {1, 2, 3},

{4, 5, 6},

{7, 8, 9} };

rotate90Clockwise(mat);

printMatrix(mat);

return 0;

}

PROG 3

#include <stdio.h>

#define N 3

int sumDiagonals(int mat[N][N]) {

int primarySum = 0, secondarySum = 0;

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

primarySum += mat[i][i];

secondarySum += mat[i][N - i - 1];

}

if (N % 2 == 1)

return primarySum + secondarySum - mat[N/2][N/2];

else

return primarySum + secondarySum;

}

int main() {

int mat[N][N] = { {1, 2, 3},

{4, 5, 6},

{7, 8, 9} };

printf("Sum of diagonals = %d\n", sumDiagonals(mat));

return 0;

}

PROG 4

#include <stdio.h>

#define ROW 2

#define COL 3

void transpose(int mat[ROW][COL], int trans[COL][ROW]) {

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

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

trans[j][i] = mat[i][j];

}}}

void printMatrix(int rows, int cols, int mat[rows][cols]) {

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

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

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

printf("\n");

}}}

int main() {

int mat[ROW][COL] = { {1, 2, 3},

{4, 5, 6} };

int trans[COL][ROW];

transpose(mat, trans);

printMatrix(COL, ROW, trans);

return 0;

}

PROG 5

#include <stdio.h>

#define ROW 3

#define COL 3

int isSparse(int mat[ROW][COL]) {

int zeroCount = 0;

int total = ROW \* COL;

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

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

if(mat[i][j] == 0)

zeroCount++;

}

}

return zeroCount > total / 2;

}

int main() {

int mat[ROW][COL] = { {0, 0, 3},

{4, 0, 6},

{0, 0, 0} };

if(isSparse(mat))

printf("Matrix is sparse\n");

else

printf("Matrix is not sparse\n");

return 0;

}

PROG 6

#include <stdio.h>

#define MAX 100

void addPolynomials(int A[], int sizeA, int B[], int sizeB, int result[], int \*resultSize) {

int maxSize = (sizeA > sizeB) ? sizeA : sizeB;

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

int coeffA = (i < sizeA) ? A[i] : 0;

int coeffB = (i < sizeB) ? B[i] : 0;

result[i] = coeffA + coeffB;

}

\*resultSize = maxSize;

}

void printPolynomial(int poly[], int size) {

for (int i = size -1; i >= 0; i--) {

if (poly[i] != 0) {

printf("%dx^%d", poly[i], i);

if (i != 0)

printf(" + ");

}

}

printf("\n");

}

int main() {

// Coefficients in ascending order of powers: constant term first

int A[] = {5, 0, 10, 6};

int B[] = {1, 2, 4};

int sizeA = sizeof(A) / sizeof(A[0]);

int sizeB = sizeof(B) / sizeof(B[0]);

int result[MAX] = {0};

int resultSize = 0;

addPolynomials(A, sizeA, B, sizeB, result, &resultSize);

printPolynomial(result, resultSize);

return 0;

}

PROG 7

#include <stdio.h>

#define MAX 100

void multiplyPolynomials(int A[], int sizeA, int B[], int sizeB, int result[]) {

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

result[i] = 0;

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

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

result[i+j] += A[i]\*B[j];

}}}

void printPolynomial(int poly[], int size) {

for(int i = size-1; i >= 0; i--) {

if(poly[i] != 0) {

printf("%dx^%d", poly[i], i);

if(i != 0)

printf(" + ");

}

}

printf("\n");

}

int main() {

int A[] = {5, 0, 10, 6};

int B[] = {1, 2, 4};

int sizeA = sizeof(A)/sizeof(A[0]);

int sizeB = sizeof(B)/sizeof(B[0]);

int result[MAX] = {0};

multiplyPolynomials(A, sizeA, B, sizeB, result);

printPolynomial(result, sizeA + sizeB -1);

return 0;

}

PROG 8

#include <stdio.h>

#include <stdlib.h>

int main() {

int n, q;

scanf("%d %d", &n, &q);

int \*\*seqList = malloc(n \* sizeof(int \*));

int \*sizes = calloc(n, sizeof(int));

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

seqList[i] = malloc(0);

int lastAnswer = 0;

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

int type, x, y;

scanf("%d %d %d", &type, &x, &y);

int idx = (x ^ lastAnswer) % n;

if(type == 1) {

sizes[idx]++;

seqList[idx] = realloc(seqList[idx], sizes[idx]\*sizeof(int));

seqList[idx][sizes[idx]-1] = y;

}

else if(type == 2) {

int val = seqList[idx][y % sizes[idx]];

lastAnswer = val;

printf("%d\n", lastAnswer);

}

}

for(int i=0; i<n; i++) free(seqList[i]);

free(seqList);

free(sizes);

return 0;

}

PROG 9

#include <stdio.h>

#define N 3

int isIdentity(int mat[N][N]) {

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

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

if(i == j && mat[i][j] != 1)

return 0;

else if(i != j && mat[i][j] != 0)

return 0;

}}

return 1;

}

int main() {

int mat[N][N] = { {1, 0, 0},

{0, 1, 0},

{0, 0, 1} };

if(isIdentity(mat))

printf("Matrix is identity\n");

else

printf("Matrix is NOT identity\n");

return 0;

}

PROG 10

#include <stdio.h>

#define ROW 4

#define COL 4

int countZeros(int mat[ROW][COL]) {

int count = 0;

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

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

if(mat[i][j] == 0)

count++;

}}

return count;

}

int main() {

int mat[ROW][COL] = {

{0, 0, 0, 1},

{0, 0, 1, 1},

{0, 1, 1, 1},

{1, 1, 1, 1}

};

printf("Number of zeros: %d\n", countZeros(mat));

return 0;

}