**CSA0358- DATA STRUCTURE OF PERFORMANCE ANALYSIS.**

**NAME: EDUKONDALU**

**REG NO: 192365010**

**DEPT: CSE (CYBER SECURITY)**

1. **ADDITION OF TWO MATRIX**

#include <stdio.h>

void addMatrices(int rows, int cols, int matrix1[rows][cols], int matrix2[rows][cols], int result[rows][cols]) {

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

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

result[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

}

int main() {

int rows = 2, cols = 2;

int matrix1[2][2] = {{1, 2}, {3, 4}};

int matrix2[2][2] = {{5, 6}, {7, 8}};

int result[2][2];

addMatrices(rows, cols, matrix1, matrix2, result);

printf("Resultant Matrix:\n");

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

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

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

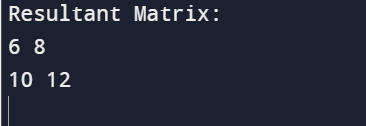
}

printf("\n");

}

return 0;

}



1. **MULTIPLICATION OF TWO MATRICES**

#include <stdio.h>

void multiplyMatrices(int firstMatrix[][3], int secondMatrix[][3], int result[][3]) {

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

result[i][j] = 0;

for (int k = 0; k < 3; ++k) {

result[i][j] += firstMatrix[i][k] \* secondMatrix[k][j];

}

}

}

}

int main() {

int firstMatrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

int secondMatrix[3][3] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};

int result[3][3];

multiplyMatrices(firstMatrix, secondMatrix, result);

printf("Resultant Matrix:\n");

for (int i = 0; i < 3; ++i) {

for (int j = 0; j < 3; ++j) {

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

if (j == 2) {

printf("\n");

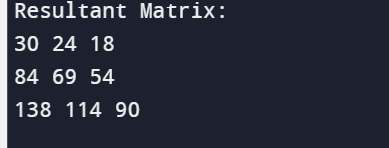
}

}

}

return 0;

}



1. **SUM OF DIAGONALS.**

#include <stdio.h>

int main() {

int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

int sum = 0;

for (int i = 0; i < 3; i++) {

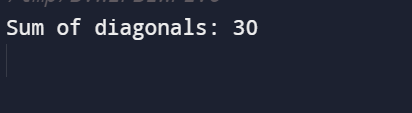
sum += matrix[i][i]; // Sum of the main diagonal

sum += matrix[i][2 - i]; // Sum of the secondary diagonal }

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

return 0;

}



**4.TRANSPOSE OF MATRIX.**

#include <stdio.h>

int main() {

int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

for (int i = 0; i < 3; i++) {

for (int j = i; j < 3; j++) {

int temp = matrix[i][j];

matrix[i][j] = matrix[j][i];

matrix[j][i] = temp;

}

}

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 3; j++) {

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

}

printf("\n");

}

return 0;

}



1. **INSERT ELEMENT IN ARRAY.**

**#include <stdio.h>**

**int main() {**

**int array[100] = {1, 2, 3, 4, 5};**

**int insertIndex = 2;**

**int newValue = 10;**

**int arraySize = 5;**

**for (int i = arraySize; i > insertIndex; i--) {**

**array[i] = array[i - 1];**

**}**

**array[insertIndex] = newValue;**

**arraySize++;**

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

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

**}**

**return 0;**

**}**

****

1. **DELETE ELEMENT IN ARRAY.**

**#include <stdio.h>**

**void deleteElement(int arr[], int size, int index) {**

**if (index < 0 || index >= size) {**

**printf("Invalid index\n");**

**return;**

**}**

**for (int i = index; i < size - 1; i++) {**

**arr[i] = arr[i + 1];**

**}**

**size--;**

**printf("Element deleted successfully\n");**

**}**

**int main() {**

**int arr[] = {1, 2, 3, 4, 5};**

**int size = sizeof(arr) / sizeof(arr[0]);**

**int index = 2;**

**printf("Original Array: ");**

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

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

**}**

**printf("\n");**

**deleteElement(arr, size, index);**

**printf("Array after deletion: ");**

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

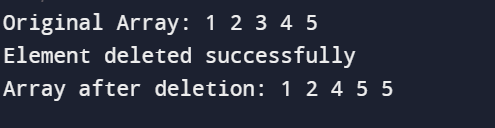
**printf("%d ", arr[i]);**

**}**

**printf("\n");**

**return 0;**

**}**

****

1. **MERGING ARRAYS**

**#include <stdio.h>**

**int main() {**

**int arr1[] = {1, 2, 3, 4, 5};**

**int size1 = sizeof(arr1) / sizeof(arr1[0]);**

**int arr2[] = {6, 7, 8, 9, 10};**

**int size2 = sizeof(arr2) / sizeof(arr2[0]);**

**int mergedSize = size1 + size2;**

**int merged[mergedSize];**

**for (int i = 0; i < size1; i++) {**

**merged[i] = arr1[i];**

**}**

**for (int i = 0; i < size2; i++) {**

**merged[size1 + i] = arr2[i];**

**}**

**printf("Merged Array: ");**

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

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

**}**

**return 0;**

**}**

****

1. **FIND DUPLICATE ELEMENT.**

**#include <stdio.h>**

**void findDuplicate(int arr[], int size) {**

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

**for (int j = i + 1; j < size; j++) {**

**if (arr[i] == arr[j]) {**

**printf("Duplicate element: %d\n", arr[j]);**

**}**

**}**

**}**

**}**

**int main() {**

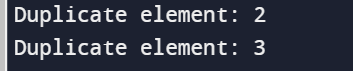
**int arr[] = {1, 2, 3, 4, 2, 5, 6, 3};**

**int size = sizeof(arr) / sizeof(arr[0]);**

**findDuplicate(arr, size);**

**return 0;**

**}**

****

1. **Display the location of given element in array and print it**

#include <stdio.h>

int main() {

int arr[] = {10, 20, 30, 40, 50};

int n = sizeof(arr) / sizeof(arr[0]);

int element = 30;

int location = -1;

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

if (arr[i] == element) {

location = i;

break;

}

}

printf("The element %d is located at index %d in the array.", element, location);

return 0;

}



1. **Ascending/Descending/Alphabet order**

#include <stdio.h>

#include <string.h>

int compareStrings(const void \*a, const void \*b) {

return strcmp(\*(const char \*\*)a, \*(const char \*\*)b);

}

int main() {

char \*words[] = {"Apple", "Banana", "Orange", "Grapes", "Kiwi"};

int numWords = sizeof(words) / sizeof(words[0]);

qsort(words, numWords, sizeof(char \*), compareStrings);

printf("Ascending Order:\n");

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

printf("%s\n", words[i]);

}

qsort(words, numWords, sizeof(char \*), compareStrings);

printf("\nDescending Order:\n");

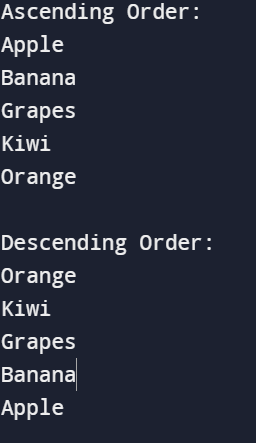
for (int i = numWords - 1; i >= 0; i--) {

printf("%s\n", words[i]);

}

return 0;

}



1. **Validation of string**

#include <stdio.h>

#include <string.h>

int isValidString(char \*str) {

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

if (!(str[i] >= 'a' && str[i] <= 'z') && !(str[i] >= 'A' && str[i] <= 'Z')) {

return 0;

}

}

return 1;

}

int main() {

char inputString[] = "ValidString123";

if (isValidString(inputString)) {

printf("The string is valid.\n");

} else {

printf("The string is not valid.\n");

}

return 0;

}



1. **Repeat twice in array.**

#include <stdio.h>

int main() {

printf("k.Edukondalu\n");

printf("192365010\n");

int array[] = {1, 2, 3, 4, 5};

int size = sizeof(array) / sizeof(array[0]);

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

for (int j = 0; j < 2; j++) {

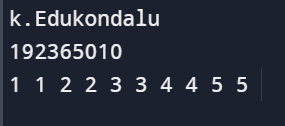
printf("%d ", array[i]);

}

}

return 0;

}



1. **Print odd and even in given array**

#include <stdio.h>

void printOddEven(int arr[], int size) {

printf("k,Edukondalu\n");

printf("192365010\n");

printf("Odd numbers: ");

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

if (arr[i] % 2 != 0) {

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

}

}

printf("\nEven numbers: ");

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

if (arr[i] % 2 == 0) {

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

}

}

}

int main() {

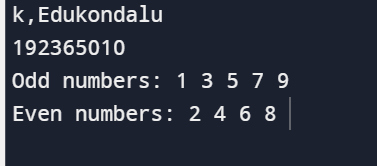
int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};

int size = sizeof(arr) / sizeof(arr[0]);

printOddEven(arr, size);

return 0;

}



1. **Sum of row and column in matrix**

**#include <stdio.h>**

**int main() {**

**printf("k.Edukondalu\n");**

**printf("192365010\n");**

**int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};**

**int row\_sum, col\_sum;**

**for (int i = 0; i < 3; i++) {**

**row\_sum = 0;**

**col\_sum = 0;**

**for (int j = 0; j < 3; j++) {**

**row\_sum += matrix[i][j];**

**col\_sum += matrix[j][i];**

**}**

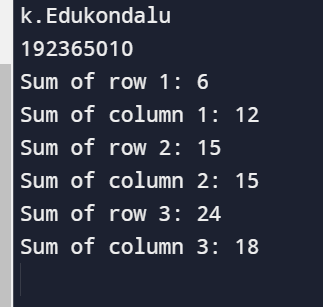
**printf("Sum of row %d: %d\n", i + 1, row\_sum);**

**printf("Sum of column %d: %d\n", i + 1, col\_sum);**

**}**

**return 0;**

**}**

****

1. **Given array elements display 5th iterated element**

#include <stdio.h>

int main() {

printf("k.Edukondalu\n");

printf("192365010\n");

int array[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int size = sizeof(array) / sizeof(array[0]);

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

if (i == 4) {

printf("5th Iterated Element: %d", array[i]);

break;

}

}

return 0;

}

