ASSIGNMENT-1

**1**. arr = []

def create():

n = int(input("Enter number of elements: "))

arr = []

for i in range(n):

val = int(input(f"Enter element {i+1}: "))

arr.append(val)

print("Array created successfully!")

def display():

if not arr:

print("Array is empty!")

else:

print("Current Array:", arr)

def insert():

if not arr:

print("Array is empty! Please create it first.")

return

pos = int(input("Enter position to insert (0-based index): "))

if pos < 0 or pos > len(arr):

print("Invalid position!")

return

val = int(input("Enter element to insert: "))

arr.insert(pos, val)

print("Element inserted successfully!")

def delete():

if not arr:

print("Array is empty! Please create it first.")

return

pos = int(input("Enter position to delete (0-based index): "))

if pos < 0 or pos >= len(arr):

print("Invalid position!")

return

deleted = arr.pop(pos)

print(f"Element {deleted} deleted successfully!")

def linear\_search():

if not arr:

print("Array is empty! Please create it first.")

return

val = int(input("Enter element to search: "))

found = False

for i in range(len(arr)):

if arr[i] == val:

print(f"Element found at position {i} (0-based index).")

found = True

break

if not found:

print("Element not found in the array.")

while True:

print("\n--- MENU ---")

print("1. CREATE\n")

print("2. DISPLAY\n")

print("3. INSERT\n")

print("4. DELETE\n")

print("5. LINEAR SEARCH\n")

print("6. EXIT")

choice = input("Enter your choice (1-6): ")

if choice == '1':

create()

elif choice == '2':

display()

elif choice == '3':

insert()

elif choice == '4':

delete()

elif choice == '5':

linear\_search()

elif choice == '6':

print("Exiting program. Goodbye!")

break

else:

print("Invalid choice! Please select between 1-6.")

**2**. Step-by-Step Approach

1. Start
2. Input the size of the array (n).
3. Read all n elements into an array arr.
4. For each element at index i from 0 to n-1:
   * Compare it with all subsequent elements (j from i+1 to n-1).
   * If a duplicate is found (arr[i] == arr[j]):
     + Shift all elements from j+1 to n-1 one position to the left.
     + Decrease n by 1 (because the array now has one fewer element).
     + Do not increment j in this case, because the new element at position j needs to be checked again.
5. End loop after processing all elements.
6. Output the new array of size n — now containing only unique elements.
7. End

**3**. Final Output- 10000

**4**.

**a. Reverse the elements of an array**

1. Start with an array of size n.
2. Use two pointers:
   * One at the start (left = 0)
   * One at the end (right = n-1)
3. Swap the elements at left and right.
4. Increment left and decrement right until they meet.

**EXAMPLE:** Input: [1, 2, 3, 4, 5]

Output: [5, 4, 3, 2, 1]

**CODE:**

arr = [1, 2, 3, 4, 5]

n = len(arr)

left, right = 0, n - 1

while left < right:

arr[left], arr[right] = arr[right], arr[left]

left += 1

right -= 1

print("Reversed array:", arr)

**b. Matrix Multiplication**

If A is of size m x p and B is of size p x n:

* The result matrix C will be of size m x n.
* Formula: C[i][j] = sum(A[i][k] \* B[k][j] for k in range(p))

**EXAMPLE:** A = [[1, 2], [3, 4]]

B = [[5, 6], [7, 8]]

C = [[(1\*5 + 2\*7), (1\*6 + 2\*8)],

[(3\*5 + 4\*7), (3\*6 + 4\*8)]]

C = [[19, 22], [43, 50]]

**CODE**: A = [[1, 2], [3, 4]]

B = [[5, 6], [7, 8]]

m, p = len(A), len(A[0])

p2, n = len(B), len(B[0])

if p != p2:

print("Matrix multiplication not possible!")

else:

C = [[0]\*n for \_ in range(m)]

for i in range(m):

for j in range(n):

for k in range(p):

C[i][j] += A[i][k] \* B[k][j]

print("Matrix multiplication result:")

for row in C:

print(row)

**c. Transpose of a Matrix**

* Given a matrix of size m x n, the transpose will be size n x m.
* transpose[i][j] = original[j][i].
* **EXAMPLE:**

Original: Transpose:

1 2 3 1 4

4 5 6 2 5

3 6

**CODE:**

matrix = [[1, 2, 3], [4, 5, 6]]

m, n = len(matrix), len(matrix[0])

transpose = [[0]\*m for \_ in range(n)]

for i in range(m):

for j in range(n):

transpose[j][i] = matrix[i][j]

print("Transpose of matrix:")

for row in transpose:

print(row)

**5.**

Input the number of rows (m) and columns (n).

Read the elements into a 2D array (matrix[m][n]).

For each row:

Initialize row\_sum = 0

Add each element in that row to row\_sum.

For each column:

Initialize col\_sum = 0

Add each element in that column to col\_sum.

Print all row sums and column sums.

**EXAMPLE:**

Matrix:

1 2 3

4 5 6

Row sums:

Row 1: 6

Row 2: 15

Column sums:

Col 1: 5

Col 2: 7

Col 3: 9

**CODE:**

#include <stdio.h>

int main() {

int m, n, i, j;

printf("Enter number of rows: ");

scanf("%d", &m);

printf("Enter number of columns: ");

scanf("%d", &n);

int matrix[m][n];

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

for (i = 0; i < m; i++) {

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

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

}

}

// Row sums

for (i = 0; i < m; i++) {

int row\_sum = 0;

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

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

}

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

}

// Column sums

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

int col\_sum = 0;

for (i = 0; i < m; i++) {

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

}

printf("Sum of column %d = %d\n", j + 1, col\_sum);

}

return 0;

**}**