UCS301 Data Structures

Lab Assignment 1

-Shefali Sharma

1024030284

2C35

Q1) Develop a Menu driven program to demonstrate the following operations of Arrays

1.CREATE 2.DISPLAY 3.INSERT 4.DELETE 5.LINEAR SEARCH 6.EXIT

A1) Aim:

*To implement a menu-driven program to demonstrate **Create, Display, Insert, Delete, Linear Search, and Exit** operations on arrays.*

```
#include <iostream>

using namespace std;


#define MAX 100

int arr[MAX], n = 0;


void create() {
   cout << "Enter number of elements: ";
   cin >> n;
   cout << "Enter " << n << " elements:\n";
   for (int i = 0; i < n; i++)
      cin >> arr[i];
```

```
}

void display() {
    if (n == 0) {
        cout << "Array is empty.\n";
        return;
    }
    cout << "Array elements: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

void insertElement() {
    int pos, val;
    cout << "Enter position (1-" << n+1 << ") to insert: ";
    cin >> pos;
    if (pos < 1 || pos > n+1) {
        cout << "Invalid position.\n";
        return;
    }
    cout << "Enter value: ";
    cin >> val;
    for (int i = n; i >= pos; i--)
        arr[i] = arr[i-1];
    arr[pos-1] = val;
```

```
    n++;
}

void deleteElement() {
    int pos;
    cout << "Enter position (1-" << n << ") to delete: ";
    cin >> pos;
    if (pos < 1 || pos > n) {
        cout << "Invalid position.\n";
        return;
    }
    for (int i = pos-1; i < n-1; i++)
        arr[i] = arr[i+1];
    n--;
}

void linearSearch() {
    int key;
    bool found = false;
    cout << "Enter element to search: ";
    cin >> key;
    for (int i = 0; i < n; i++) {
        if (arr[i] == key) {
            cout << "Element found at position " << i+1 << ".\n";
            found = true;
            break;
```

```
        }
    }
    if (!found)
        cout << "Element not found.\n";
}

int main() {
    int choice;
    do {
        cout << "\n---- MENU ----\n";
        cout << "1.CREATE\n2.DISPLAY\n3.INSERT\n4.DELETE\n5.LINEAR
SEARCH\n6.EXIT\n";
        cout << "Enter choice: ";
        cin >> choice;
        switch (choice) {
            case 1: create(); break;
            case 2: display(); break;
            case 3: insertElement(); break;
            case 4: deleteElement(); break;
            case 5: linearSearch(); break;
            case 6: cout << "Exiting...\n"; break;
            default: cout << "Invalid choice.\n";
        }
    } while (choice != 6);
    return 0;
}
```

Q2) Design the logic to remove the duplicate elements from an Array and after the deletion the array should contain the unique elements.

A2)#include <iostream>

```
using namespace std;


int main() {
   int n;
   cout << "Enter number of elements: ";
   cin >> n;
   int arr[n];
   cout << "Enter " << n << " elements:\n";
   for (int i = 0; i < n; i++)
      cin >> arr[i];

   for (int i = 0; i < n; i++) {
      for (int j = i+1; j < n; j++) {
         if (arr[i] == arr[j]) {
            for (int k = j; k < n-1; k++)
               arr[k] = arr[k+1];
            n--;
            j--;
         }
      }
   }

   cout << "Array after removing duplicates: ";
```

```
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

Q3)Predict the Output of the following program

```
int main()
{
int i;
int arr[5] = {1};
for (i = 0; i < 5; i++)
printf("%d",arr[i]);
return 0;
}
```

A3)

```
#include <iostream>
using namespace std;

int main() {
    int i;
    int arr[5] = {1};
    for (i = 0; i < 5; i++)
        cout << arr[i];
    return 0;
}
```

Prediction:
Only arr[0] is initialized to 1, others are default-initialized to 0.

Output: 10000

Q4) Implement the logic to

a. Reverse the elements of an array

b. Find the matrix multiplication

c. Find the Transpose of a Matrix

A4 (a)) Reverse an Array

```
#include <iostream>
using namespace std;

int main() {
   int n, temp;
   cout << "Enter number of elements: ";
   cin >> n;
   int arr[n];
   cout << "Enter elements:\n";
   for (int i = 0; i < n; i++)
      cin >> arr[i];

   for (int i = 0; i < n/2; i++) {
      temp = arr[i];
      arr[i] = arr[n-i-1];
```

```
        arr[n-i-1] = temp;
    }

    cout << "Reversed array: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    return 0;
}
```

Q4b) Matrix Multiplication

```
#include <iostream>
using namespace std;

int main() {
    int r1, c1, r2, c2;
    cout << "Enter rows and cols of first matrix: ";
    cin >> r1 >> c1;
    cout << "Enter rows and cols of second matrix: ";
    cin >> r2 >> c2;

    if (c1 != r2) {
        cout << "Matrix multiplication not possible.\n";
        return 0;
    }

    int a[r1][c1], b[r2][c2], res[r1][c2] = {0};
```

```
    cout << "Enter first matrix:\n";
    for (int i = 0; i < r1; i++)
        for (int j = 0; j < c1; j++)
            cin >> a[i][j];

    cout << "Enter second matrix:\n";
    for (int i = 0; i < r2; i++)
        for (int j = 0; j < c2; j++)
            cin >> b[i][j];

    for (int i = 0; i < r1; i++)
        for (int j = 0; j < c2; j++)
            for (int k = 0; k < c1; k++)
                res[i][j] += a[i][k] * b[k][j];

    cout << "Result matrix:\n";
    for (int i = 0; i < r1; i++) {
        for (int j = 0; j < c2; j++)
            cout << res[i][j] << " ";
        cout << endl;
    }
    return 0;
}
```

Q4 C) Transpose of a Matrix

```
#include <iostream>
using namespace std;

int main() {
    int r, c;
    cout << "Enter rows and columns: ";
    cin >> r >> c;
    int a[r][c], trans[c][r];

    cout << "Enter matrix:\n";
    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            cin >> a[i][j];

    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            trans[j][i] = a[i][j];

    cout << "Transpose:\n";
    for (int i = 0; i < c; i++) {
        for (int j = 0; j < r; j++)
            cout << trans[i][j] << " ";
        cout << endl;
    }
    return 0;
}
```

Q5) Write a program to find sum of every row and every column in a two-dimensional array.

A5) Sum of Every Row and Column in a 2D Array:

```
#include <iostream>
using namespace std;

int main() {
   int r, c, sum;
   cout << "Enter rows and columns: ";
   cin >> r >> c;
   int a[r][c];

   cout << "Enter matrix elements:\n";
   for (int i = 0; i < r; i++)
      for (int j = 0; j < c; j++)
         cin >> a[i][j];

   for (int i = 0; i < r; i++) {
      sum = 0;
      for (int j = 0; j < c; j++)
         sum += a[i][j];
```

```
        cout << "Sum of row " << i+1 << " = " << sum << endl;
    }

    for (int j = 0; j < c; j++) {
        sum = 0;
        for (int i = 0; i < r; i++)
            sum += a[i][j];
        cout << "Sum of column " << j+1 << " = " << sum << endl;
    }
    return 0;
}
```