**Practice Assignment – Part 2**

// Q1. butterfly pattern

#include<iostream>

using namespace std;

int main() {

    int n;

    cout<<"Enter the number of rows for each half of the butterfly pattern: ";

    cin>>n;

    for (int i=1;i<=n;i++) {

        for (int j =1;j<=i;j++) {

            cout<<"\*";

        }

        int space=2\*n-2\*i;

        for (int j=1;j<=space;j++) {

            cout<<" ";

        }

        for (int j=1;j<=i;j++) {

            cout<<"\*";

        }

        cout<<endl;

    }

    for (int i=n;i>=1;i--) {

        for (int j=1;j<=i;j++) {

            cout<<"\*";

        }

        int space=2\*n-2\*i;

        for (int j=1;j<=space;j++) {

            cout<<" ";

        }

        for (int j=1;j<=i;j++) {

            cout<<"\*";

        }

        cout<<endl;

    }

    return 0;

}

// Q2. array reverse

#include<iostream>

using namespace std;

int main() {

    int n;

    cout<<"Enter the number of elements: ";

    cin>>n;

    int arr[n];

    for (int i=0;i<n;i++) {

        cout <<"Enter element "<<i+1<<" : ";

        cin>>arr[i];

    }

    cout<<"Array in reverse order: ";

    for (int i=n-1;i>=0;i--) {

        cout<<arr[i]<<" ";

    }

    cout<<endl;

    return 0;

}

// Q3. find maximum and minimum

#include<iostream>

using namespace std;

int main(){

    int n;

    cout<<"Enter the number of elements in array: ";

    cin>>n;

    int arr[n];

    for(int i=0;i<n;i++){

        cout<<"Enter element "<<i+1<<" : ";

        cin>>arr[i];

    }

    int minElement=arr[0];

    int maxElement=arr[0];

    for (int i = 1; i < n; i++) {

        if (arr[i] < minElement) {

            minElement = arr[i];

        }

        if (arr[i] > maxElement) {

            maxElement = arr[i];

        }

    }

    cout << "Minimum element in the array: " << minElement << endl;

    cout << "Maximum element in the array: " << maxElement << endl;

    return 0;

}

// Q4. Sum and average

#include<iostream>

using namespace std;

int main(){

    int n;

    cout<<"Enter the number of elements in array: ";

    cin>>n;

    int arr[n];

    for(int i=0;i<n;i++){

        cout<<"Enter element "<<i+1<<" : ";

        cin>>arr[i];

    }

    int sum=0;

    for(int i=0;i<n;i++){

        sum+=arr[i];

    }

    cout<<"The sum of all elements in array is: "<<sum<<endl;

    cout<<"The average of all elements in array is: "<<sum/n<<endl;

    return 0;

}

// Q5. sort (logic)

/\*

1. Compare Adjacent Elements:

Compare each pair of adjacent elements (like arr[i] and arr[i+1]).

2. Swap if Necessary:

If the element at index i is greater than the element at index i+1, swap them.

3. Repeat the Process:

After each pass through the array, the largest unsorted element "bubbles up" to its correct position at the end of the array. The algorithm continues to loop through the array, reducing the range of comparison each time.

4. Stop when No Swaps are Made:

If a full pass through the array is completed without any swaps, the array is already sorted, and the algorithm stops early.

\*/

// Q6. linear search(binary search)

#include<iostream>

using namespace std;

int main(){

    int n;

    cout<<"Enter the number of elements in array: ";

    cin>>n;

    int arr[n];

    for(int i=0;i<n;i++){

        cout<<"Enter element "<<i+1<<" : ";

        cin>>arr[i];

    }

    int searchElement,found=-1;

    cout<<"Enter the element to be searched: ";

    cin>>searchElement;

    for (int i=0;i<n;i++) {

        if (arr[i] == searchElement){

            found = i;

            break;

        }

    }

    if (found != -1) {

        cout << "Element " << searchElement << " found at index " << found << "." << endl;

    } else {

        cout << "Element " << searchElement << " not found in the array." << endl;

    }

    return 0;

}

// Q7. remove duplicated

#include<iostream>

using namespace std;

int main(){

    int n;

    cout<<"Enter number of elements in array: ";

    cin>>n;

    int arr[n];

    for(int i=0;i<n;i++){

        cout<<"Enter element "<<i+1<<" : ";

        cin>>arr[i];

    }

    int newSize = n;

    for(int i=0;i<newSize;i++) {

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

            if(arr[i] == arr[j]) {

                for(int k=j;k<newSize-1;k++) {

                    arr[k] = arr[k + 1];

                }

                newSize--;

                j--;

            }

        }

    }

    cout << "Array after removing duplicates: ";

    for(int i=0;i<newSize;i++) {

        cout << arr[i] << " ";

    }

    cout << endl;

    return 0;

}

// Q8. long consecutive subsequence

#include<iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter number of elements in the array: ";

    cin >> n;

    int arr[n];

    for (int i = 0; i < n; i++) {

        cout << "Enter the element "<<i+1<< " : ";

        cin >> arr[i];

    }

    for (int i = 0; i < n - 1; i++) {

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                int temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

            }

        }

    }

    int longestStreak = 1, currentStreak = 1;

    for (int i = 1; i < n; i++) {

        if (arr[i] == arr[i - 1] + 1) {

            currentStreak++;

        } else if (arr[i] != arr[i - 1]) {

            longestStreak = max(longestStreak, currentStreak);

            currentStreak = 1;

        }

    }

    longestStreak = max(longestStreak, currentStreak);

    cout << "Length of longest consecutive subsequence: " << longestStreak << endl;

    return 0;

}

// Q9. matrix transpose

#include<iostream>

using namespace std;

int main(){

    int row,col;

    cout<<"Enter number of rows and column: ";

    cin>>row>>col;

    int matrix[row][col],transpose[col][row];

    for(int i=0;i<row;i++){

        for(int j=0;j<col;j++){

            cout<<"Enter element at position "<<"["<<i<<" ]"<<" ["<<j<<"]";

            cin>>matrix[i][j];

        }

    }

    for (int i = 0; i < row; i++) {

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

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

        }

    }

    cout << "\nOriginal Matrix:\n";

    for (int i = 0; i < row; i++) {

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

            cout << matrix[i][j] << " ";

        }

        cout << endl;

    }

    cout << "\nTranspose of the Matrix:\n";

    for (int i = 0; i < col; i++) {

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

            cout << transpose[i][j] << " ";

        }

        cout << endl;

    }

    return 0;

}

// Q10. matrix multiplication

#include<iostream>

using namespace std;

int main() {

    int m, n, p;

    cout << "Enter number of rows and columns for first matrix: ";

    cin >> m >> n;

    cout << "Enter number of rows and columns for second matrix: ";

    cin >> n >> p;

    int A[m][n], B[n][p], C[m][p];

    cout << "Enter elements of first matrix (A):\n";

    for (int i = 0; i < m; i++) {

        for (int j = 0; j < n; j++) {

            cout << "Enter element at position A[" << i + 1 << "][" << j + 1 << "]: ";

            cin >> A[i][j];

        }

    }

    cout << "Enter elements of second matrix (B):\n";

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < p; j++) {

            cout << "Enter element at position B[" << i + 1 << "][" << j + 1 << "]: ";

            cin >> B[i][j];

        }

    }

    for (int i = 0; i < m; i++) {

        for (int j = 0; j < p; j++) {

            C[i][j] = 0;

        }

    }

    for (int i = 0; i < m; i++) {

        for (int j = 0; j < p; j++) {

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

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

            }

        }

    }

    cout << "\nResultant Matrix (C) after multiplication:\n";

    for (int i = 0; i < m; i++) {

        for (int j = 0; j < p; j++) {

            cout << C[i][j] << " ";

        }

        cout << endl;

    }

    return 0;

}

// Q11. diagonal sum of a matrix

#include<iostream>

using namespace std;

int main() {

    int n;

    cout << "Enter the size of the square matrix: ";

    cin >> n;

    int matrix[n][n];

    int primaryDiagonal=0, secondaryDiagonal=0;

    cout << "Enter elements of the matrix: ";

    for (int i=0;i<n;i++) {

        for (int j=0;j<n;j++) {

            cout<<"Enter element at position A["<<i+1<<"]["<<j+1<<"]: ";

            cin>>matrix[i][j];

        }

    }

    for (int i = 0; i < n; i++) {

        primaryDiagonal += matrix[i][i];

        secondaryDiagonal += matrix[i][n - i - 1];

    }

    cout << "\nPrimary Diagonal Sum: " << primaryDiagonal << endl;

    cout << "Secondary Diagonal Sum: " << secondaryDiagonal << endl;

    return 0;

}

// Q12. find row in a matrix which has maximum sum of its elements

#include<iostream>

using namespace std;

int main() {

    int rows, cols;

    cout << "Enter number of rows: ";

    cin >> rows;

    cout << "Enter number of columns: ";

    cin >> cols;

    int matrix[rows][cols];

    int maxRowSum = -1;

    int rowWithMaxSum = -1;

    cout << "Enter elements of the matrix:\n";

    for (int i = 0; i < rows; i++) {

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

            cout << "Enter element at position A[" << i + 1 << "][" << j + 1 << "]: ";

            cin >> matrix[i][j];

        }

    }

    for (int i = 0; i < rows; i++) {

        int rowSum = 0;

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

            rowSum += matrix[i][j];

        }

        if (rowSum > maxRowSum) {

            maxRowSum = rowSum;

            rowWithMaxSum = i;

        }

    }

    cout << "\nRow with the maximum sum is Row " << rowWithMaxSum + 1 << " with a sum of " << maxRowSum << endl;

    return 0;

}

// Q13. search in a sorted matrix

#include<iostream>

using namespace std;

int main() {

    int rows, cols, target;

    cout << "Enter number of rows: ";

    cin >> rows;

    cout << "Enter number of columns: ";

    cin >> cols;

    int matrix[rows][cols];

    cout << "Enter elements of the matrix: ";

    for (int i=0;i<rows;i++) {

        for (int j=0;j<cols;j++) {

            cin>>matrix[i][j];

        }

    }

    cout << "Enter target value to search: ";

    cin >> target;

    int row = 0, col = cols - 1;

    bool found = false;

    while (row < rows && col >= 0) {

        if (matrix[row][col] == target) {

            cout << "Target " << target << " found at position (" << row + 1 << ", " << col + 1 << ").\n";

            found = true;

            break;

        }

        else if (matrix[row][col] > target) {

            col--;

        }

        else {

            row++;

        }

    }

    if (!found) {

        cout << "Target " << target << " not found in the matrix.\n";

    }

    return 0;

}

// Q14. boundary elements of a matrix

#include<iostream>

using namespace std;

int main() {

    int rows, cols;

    cout << "Enter number of rows: ";

    cin >> rows;

    cout << "Enter number of columns: ";

    cin >> cols;

    int matrix[rows][cols];

    int maxRowSum = -1;

    int rowWithMaxSum = -1;

    cout << "Enter elements of the matrix:\n";

    for (int i = 0; i < rows; i++) {

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

            cout << "Enter element at position [" << i + 1 << "][" << j + 1 << "]: ";

            cin >> matrix[i][j];

        }

    }

    for (int i = 0; i < rows; i++) {

        int rowSum = 0;

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

            rowSum += matrix[i][j];

        }

        if (rowSum > maxRowSum) {

            maxRowSum = rowSum;

            rowWithMaxSum = i;

        }

    }

    cout << "\nRow with the maximum sum is Row " << rowWithMaxSum + 1 << " with a sum of " << maxRowSum << endl;

    return 0;

}

// Q15. find saddle point

#include<iostream>

using namespace std;

int main() {

    int rows, cols;

    cout << "Enter number of rows: ";

    cin >> rows;

    cout << "Enter number of columns: ";

    cin >> cols;

    int matrix[rows][cols];

    cout << "Enter elements of the matrix:\n";

    for (int i = 0; i < rows; i++) {

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

            cin >> matrix[i][j];

        }

    }

    bool foundSaddlePoint = false;

    for (int i = 0; i < rows; i++) {

        int minElement = matrix[i][0];

        int minIndex = 0;

        for (int j = 1; j < cols; j++) {

            if (matrix[i][j] < minElement) {

                minElement = matrix[i][j];

                minIndex = j;

            }

        }

        bool isSaddlePoint = true;

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

            if (matrix[k][minIndex] > minElement) {

                isSaddlePoint = false;

                break;

            }

        }

        if (isSaddlePoint) {

            cout << "Saddle Point is: " << minElement << " at position (" << i + 1 << ", " << minIndex + 1 << ")\n";

            foundSaddlePoint = true;

        }

    }

    if (!foundSaddlePoint) {

        cout << "No Saddle Point is found in the matrix.\n";

    }

    return 0;

}