C Programs for Practice (Arrays)

1. **C Program to determine the largest element in an Array**

```
#include <stdio.h>

int findMax(int arr[], int n) {

    // Assume the first element is the largest
    int max = arr[0];
    for (int i = 1; i < n; i++) {

        // Update max if arr[i] is greater
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

int main() {
    int arr[] = {5, 2, 7, 6};
    int n = sizeof(arr) / sizeof(arr[0]);

    printf("%d\n", findMax(arr, n));
    return 0;
}
```

2. **C Program to determine the maximum and minimum element of an Array**

```
/* structure is used to return two values from minMax() */
#include<stdio.h>
struct pair
{
int min;
int max;
};

struct pair getMinMax(int arr[], int n)
{
struct pair minmax;
int i;

/*If there is only one element then return it as min and max both*/
if (n == 1)
```

```
{
        minmax.max = arr[0];
        minmax.min = arr[0];
        return minmax;
}

/* If there are more than one elements, then initialize min
        and max*/
if (arr[0] > arr[1])
{
        minmax.max = arr[0];
        minmax.min = arr[1];
}
else
{
        minmax.max = arr[1];
        minmax.min = arr[0];
}

for (i = 2; i<n; i++)
{
        if (arr[i] > minmax.max)
        minmax.max = arr[i];

        else if (arr[i] < minmax.min)
        minmax.min = arr[i];
}

return minmax;
}

/* Driver program to test above function */
int main()
{
int arr[] = {1000, 11, 445, 1, 330, 3000};
int arr_size = 6;
struct pair minmax = getMinMax (arr, arr_size);
printf("nMinimum element is %d", minmax.min);
printf("nMaximum element is %d", minmax.max);
getchar();
}
```

3. **C Program to calculate the average of all elements present in an Array**

```
#include <stdio.h>
```

```
float getAvg(int arr[], int n) {
    int sum = 0;

    // Find the sum of all elements
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }

     // Return the average
    return (float)sum / n;
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int n = sizeof(arr) / sizeof(arr[0]);

     // Calculate the average of array arr
    float res = getAvg(arr, n);

    printf("%.2f\n", res);
    return 0;
}
```

4. C Program to sort the elements of an Array in Ascending order

```
#include <stdio.h>
#include <stdlib.h>

// Custom comparator
int comp(const void* a, const void* b) {

     // If a is smaller, positive value will be returned
    return (*(int*)a - *(int*)b);
}

int main() {
    int arr[] = { 2 ,6, 1, 5, 3, 4 };
    int n = sizeof(arr) / sizeof(arr[0]);

    // Sort the array using qsort
    qsort(arr, n, sizeof(int), comp);

    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
```

```
    return 0;
}
```

5. **C Program to sort the elements of an Array in Descending order**

```
#include <stdio.h>
#include <stdlib.h>

// Comparator function to sort in descending order
int comp(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
}

int main() {
    int arr[] = {1, 3, 5, 4, 2};
    int n = sizeof(arr) / sizeof(arr[0]);

    // Sorting the array using qsort
    qsort(arr, n, sizeof(int), comp);

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

6. **C Program to merge two arrays**

```
// C program to merge two arrays into a new array using
// memcpy()
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int* mergeArrays(int arr1[], int n1, int arr2[], int n2) {

    // Resultant array to store merged array
    int *res = (int*)malloc(sizeof(int) * n1 * n2);

    // Copy elements of the first array
    memcpy(res, arr1, n1 * sizeof(int));

    // Copy elements of the second array
    memcpy(res + n1, arr2, n2 * sizeof(int));

    return res;
```

```
}

int main() {
    int arr1[] = {1, 3, 5};
    int arr2[] = {2, 4, 6};
    int n1 = sizeof(arr1) / sizeof(arr1[0]);
    int n2 = sizeof(arr2) / sizeof(arr2[0]);

     // Merge arr1 and arr2
    int* res = mergeArrays(arr1, n1, arr2, n2);

    for (int i = 0; i < n1 + n2; i++)
        printf("%d ", res[i]);

    return 0;
}
```

7. C Program to copy all elements of one array into another array

```
#include <stdio.h>
#include <string.h>

int main() {
    int arr1[] = {1, 2, 3, 4, 5};
    int n = sizeof(arr1) / sizeof(arr1[0]);
    int arr2[n];

    // Use memcpy to copy arr1 to arr2
    memcpy(arr2, arr1, n * sizeof(arr1[0]));

    for (int i = 0; i < n; i++)
        printf("%d ", arr2[i]);
    return 0;
}
```

8. C Program to check whether two matrices are equal or not

```
// C Program to check if two matrices are equals or not
#include <stdio.h>
#define N 4 // Macros

// This function returns 1 if A[][] and B[][] are equal
// otherwise it returns 0
int areSame(int A[][N], int B[][N])
{
        int i, j;
```

```
        for (i = 0; i < N; i++)
                for (j = 0; j < N; j++)
                        if (A[i][j] != B[i][j])
                                return 0;
        return 1;
}

// driver code
int main()
{

        // create two matrices
        int A[N][N] = { { 1, 2, 3, 4 },
                                        { 1, 2, 3, 4 },
                                        { 1, 2, 3, 4 },
                                        { 1, 2, 3, 4 } };

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

        // Display first matrix
        printf("\n First Matrix \n");
        for (int i = 0; i < N; i++) {
                for (int j = 0; j < N; j++) {
                        printf(" %d", A[i][j]);
                }
                printf("\n");
        }

        // Display second matrix
        printf("\n Second Matrix \n");
        for (int i = 0; i < N; i++) {
                for (int j = 0; j < N; j++) {
                        printf(" %d", B[i][j]);
                }
                printf("\n");
        }

        // check if Matrices are equals or not
        // areSame(A, B) function accept two matrices as input
        // and return 1 if matrices are equals otherwise return
        // 0
```

```
        if (areSame(A, B))
                printf("\n Matrices are equal");
        else
                printf("\n Matrices are not equal");
        return 0;
}
```

9. C Program to find the transpose of a Matrix

```
// C Program to find transpose
// of a square matrix
#include <stdio.h>
#define N 4

// This function stores transpose
// of A[][] in B[][]
void transpose(int A[][N], int B[][N])
{
        int i, j;
        for (i = 0; i < N; i++)
                for (j = 0; j < N; j++)
                        // Assigns the transpose of element A[j][i] to
                        // B[i][j]
                        B[i][j] = A[j][i];
}

// Driver code
int main()
{
        int A[N][N] = { { 1, 1, 1, 1 },
                                        { 2, 2, 2, 2 },
                                        { 3, 3, 3, 3 },
                                        { 4, 4, 4, 4 } };

        int B[N][N], i, j;

        transpose(A, B);

        printf("Result matrix is \n");
        for (i = 0; i < N; i++) {
                for (j = 0; j < N; j++)
                        printf("%d ", B[i][j]);
                printf("\n");
        }
```

```
        return 0;
}
```

10. C Program to find the determinant of a Matrix

```
// C program to find Determinant
// of a matrix
#include <stdio.h>

// Dimension of input square matrix
#define N 4

// Function to get cofactor of mat[p][q]
// in temp[][]. n is current dimension
// of mat[][]
void getCofactor(int mat[N][N], int temp[N][N],
                                int p, int q, int n)
{
        int i = 0, j = 0;

        // Looping for each element of the matrix
        for (int row = 0; row < n; row++)
        {
                for (int col = 0; col < n; col++)
                {
                        // Copying into temporary matrix
                        // only those element which are
                        // not in given row and column
                        if (row != p && col != q)
                        {
                                temp[i][j++] = mat[row][col];

                                // Row is filled, so increase row
                                // index and reset col index
                                if (j == n - 1)
                                {
                                        j = 0;
                                        i++;
                                }
                        }
                }
        }
}

/* Recursive function for finding the
```

```
determinant of matrix. n is current
dimension of mat[][]. */
int determinantOfMatrix(int mat[N][N], int n)
{
	// Initialize result
	int D = 0;

	// Base case : if matrix contains
	// single element
	if (n == 1)
		return mat[0][0];

	// To store cofactors
	int temp[N][N];

	// To store sign multiplier
	int sign = 1;

	// Iterate for each element of
	// first row
	for (int f = 0; f < n; f++)
	{
		// Getting Cofactor of mat[0][f]
		getCofactor(mat, temp, 0, f, n);
		D += sign * mat[0][f]
			* determinantOfMatrix(temp, n - 1);

		// Terms are to be added with alternate sign
		sign = -sign;
	}

	return D;
}

// Function for displaying the matrix
void display(int mat[N][N],
			int row, int col)
{
	for (int i = 0; i < row; i++)
	{
		for (int j = 0; j < col; j++)
			printf(" %d", mat[i][j]);
		printf("n");
	}
```

```
}

// Driver code
int main()
{
        int mat[N][N] = {{1, 0, 2, -1},
                                        {3, 0, 0, 5},
                                        {2, 1, 4, -3},
                                        {1, 0, 5, 0}};

        // Function call
        printf("Determinant of the matrix is : %d",
                        determinantOfMatrix(mat, N));
        return 0;
}
```

11. C Program to add two matrices

```
// C program to implement
// matrix addition
#include <stdio.h>
#define N 4

// This function adds A[][] and B[][],
// and stores the result in C[][]
void add(int A[][N], int B[][N], int C[][N])
{
   int i, j;
   for (i = 0; i < N; i++)
      for (j = 0; j < N; j++)
         C[i][j] = A[i][j] + B[i][j];
}

// This function prints the matrix
void printmatrix(int D[][N])
{
   int i, j;
   for (i = 0; i < N; i++) {
      for (j = 0; j < N; j++)
         printf("%d ", D[i][j]);
      printf("\n");
   }
}

// Driver code
```

```
int main()
{
   int A[N][N] = { { 1, 1, 1, 1 },
                   { 2, 2, 2, 2 },
                   { 3, 3, 3, 3 },
                   { 4, 4, 4, 4 } };

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

   // To store result
   int C[N][N];
   int i, j;

   printf("Matrix A is \n");
   printmatrix(A);

   printf("Matrix B is \n");
   printmatrix(B);

   add(A, B, C);

   printf("Result matrix is \n");
   printmatrix(C);

   return 0;
}
```

12. C Program to multiply two matrices

```
// C program to multiply two matrices
#include <stdio.h>
#include <stdlib.h>

// matrix dimensions so that we dont have to pass them as
// parametersmat1[R1][C1] and mat2[R2][C2]
#define R1 2 // number of rows in Matrix-1
#define C1 2 // number of columns in Matrix-1
#define R2 2 // number of rows in Matrix-2
#define C2 3 // number of columns in Matrix-2

void multiplyMatrix(int m1[][C1], int m2[][C2])
{
        int result[R1][C2];
```

```
        printf("Resultant Matrix is:\n");

        for (int i = 0; i < R1; i++) {
                for (int j = 0; j < C2; j++) {
                        result[i][j] = 0;

                        for (int k = 0; k < R2; k++) {
                                result[i][j] += m1[i][k] * m2[k][j];
                        }

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

                printf("\n");
        }
}

// Driver code
int main()
{
        // R1 = 4, C1 = 4 and R2 = 4, C2 = 4 (Update these
        // values in MACROs)
        int m1[R1][C1] = { { 1, 1 }, { 2, 2 } };

        int m2[R2][C2] = { { 1, 1, 1 }, { 2, 2, 2 } };

        // if coloumn of m1 not equal to rows of m2
        if (C1 != R2) {
                printf("The number of columns in Matrix-1 must be "
                        "equal to the number of rows in "
                        "Matrix-2\n");
                printf("Please update MACROs value according to "
                        "your array dimension in "
                        "#define section\n");

                exit(EXIT_FAILURE);
        }

        // Function call
        multiplyMatrix(m1, m2);

        return 0;
}
```