Submitted by:

NAME: Sumaiya Islam

REGISTRATION NO.:23201161

SECTION:C(2)

COURSE CODE:CSE108

Submitted to:

Zaima Sartaj Taheri

Lecturer in UAP

1. Given an array of size **N.** The task is to find the maximum and the minimum element of the array using the minimum number of comparisons

ANSWER:

```
#include <stdio.h>


void findMinMax(int arr[], int N, int *min, int *max) {
    *min = arr[0];
    *max = arr[0];


    for (int i = 1; i < N; i++) {
        if (arr[i] < *min) *min = arr[i];
        if (arr[i] > *max) *max = arr[i];
    }
}


int main() {
    int N;


    printf("Enter the number of elements in the array: ");
    scanf("%d", &N);


    if (N <= 0) {
        printf("Array size must be positive.\n");
        return 1;
    }


    int arr[N];
    printf("Enter %d elements:\n", N);
    for (int i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }

    int min, max;
    findMinMax(arr, N, &min, &max);
    printf("Minimum element is: %d\nMaximum element is: %d\n", min, max);

    return 0;
}
```

2. Given an array **arr[]**, the task is to **reverse** the array. Reversing an array means **rearranging** the elements such that the **first** element becomes the **last**, the **second** element becomes **second last** and so on.

Answer:

```
#include <stdio.h>

void reverseArray(int arr[], int n) {
    int start = 0;
    int end = n - 1;

    while (start < end) {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;

        start++;
        end--;
    }
}

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

    printf("Original array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    reverseArray(arr, n);

    printf("Reversed array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

3. Given an array, the task is to cyclically rotate the array clockwise by one time.

ANSWER:

```
#include <stdio.h>

void rotateArrayByOne(int arr[], int n) {
    int lastElement = arr[n - 1];

    for (int i = n - 1; i > 0; i--) {
        arr[i] = arr[i - 1];
    }

    arr[0] = lastElement;
}

void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

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

    printf("Original array: ");
    printArray(arr, n);

    rotateArrayByOne(arr, n);

    printf("Rotated array: ");
    printArray(arr, n);

    return 0;
}
```

4. **Sorting** an array means arranging the elements of the array in a certain order. Generally sorting in an array is done to arrange the elements in increasing or decreasing order.
Problem statement: Given an array of integers **arr**, the task is to sort the array in ascending order and return it, **without using any built-in** functions.

ANSWER:

```
#include <stdio.h>


void bubbleSort(int arr[], int n) {
    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;
            }
        }
    }

}


void printArray(int arr[], int n) {

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

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

    }

    printf("\n");

}


int main() {

    int arr[] = {64, 34, 25, 12, 22, 11, 90};

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


    printf("Original array: ");

    printArray(arr, n);


    bubbleSort(arr, n);


    printf("Sorted array: ");

    printArray(arr, n);


    return 0;     }
```

5.Given an array of n integers. The task is to print the duplicates in the given array. If there are no duplicates then print -1.

Answer:

```
#include <stdio.h>

void printDuplicates(int arr[], int n) {
    int found = 0;
    int visited[n];
    for (int i = 0; i < n; i++) {
        visited[i] = 0;
    }

    for (int i = 0; i < n; i++) {
        if (visited[i] == 1) {
            continue;
        }
        int count = 1;
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                count++;
                visited[j] = 1;
            }
        }
        if (count > 1) {
            printf("%d ", arr[i]);
            found = 1;
        }
    }

    if (found == 0) {
        printf("-1");
    }
}

int main() {
    int arr1[] = {2, 10, 10, 100, 2, 10, 11, 2, 11, 2};
    int n1 = sizeof(arr1) / sizeof(arr1[0]);
    printf("Output for first array: ");
    printDuplicates(arr1, n1);

    printf("\n");

    int arr2[] = {5, 40, 1, 40, 100000, 1, 5, 1};
    int n2 = sizeof(arr2) / sizeof(arr2[0]);
    printf("Output for second array: ");
    printDuplicates(arr2, n2);


    return 0;
}
```

6. Given a sorted array **arr[]** of size N and a number **X**, you need to find the number of occurrences of **X** in given array.

Answer:

```
#include <stdio.h>


int findFirstOccurrence(int arr[], int N, int X) {
    int left = 0, right = N - 1;
    int firstIndex = -1;


    while (left <= right) {
        int mid = left + (right - left) / 2;


        if (arr[mid] == X) {
            firstIndex = mid;
            right = mid - 1;
        } else if (arr[mid] < X) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }
    return firstIndex;
}

int findLastOccurrence(int arr[], int N, int X) {
    int left = 0, right = N - 1;
    int lastIndex = -1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == X) {
            lastIndex = mid;
            left = mid + 1;
        } else if (arr[mid] < X) {
            left = mid + 1;
        } else {
            right = mid - 1;
        }
    }

    return lastIndex;
}

int countOccurrences(int arr[], int N, int X) {
    int firstIndex = findFirstOccurrence(arr, N, X);
    if (firstIndex == -1) {
        return 0;
    }

    int lastIndex = findLastOccurrence(arr, N, X);
    return lastIndex - firstIndex + 1;
}

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

    int X = 2;
    printf("Count of %d in the array: %d\n", X, countOccurrences(arr, N, X));

    X = 4;
    printf("Count of %d in the array: %d\n", X, countOccurrences(arr, N, X));

    return 0;
}
```

7. sort the array of 0s,1s and 2s.

Answer:

```
#include <stdio.h>

void sortArray(int arr[], int n) {
    int low = 0, mid = 0, high = n - 1;

    while (mid <= high) {
        if (arr[mid] == 0) {
            int temp = arr[low];
            arr[low] = arr[mid];
            arr[mid] = temp;
            low++;
            mid++;
        } else if (arr[mid] == 1) {
            mid++;
        } else {
            int temp = arr[mid];
            arr[mid] = arr[high];
            arr[high] = temp;
            high--;
        }
    }
}

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

    sortArray(arr, n);

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

8. An array contains both positive and negative numbers in random order. Rearrange the array elements so that all negative numbers appear before all positive numbers.

Answer:

```
#include <stdio.h>


void rearrange(int arr[], int size) {
    int left = 0, right = size - 1;

    while (left < right) {
        while (arr[left] < 0 && left < right) {
            left++;
        }

        while (arr[right] >= 0 && left < right) {
            right--;
        }

        if (left < right) {
            int temp = arr[left];
            arr[left] = arr[right];
            arr[right] = temp;
        }
    }
}
```

```
int main() {
    int arr[] = {-12, 11, -13, -5, 6, -7, 5, -3, -6};
    int size = sizeof(arr) / sizeof(arr[0]);

    rearrange(arr, size);

    printf("Rearranged array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

9. Given a **binary** 2D array, where each row is **sorted**. Find the row with the maximum number of 1s.

Answer:

```
#include <stdio.h>

int findRowWithMaxOnes(int arr[][4], int rows) {
    int maxRowIndex = -1;
    int maxCount = 0;

    for (int i = 0; i < rows; i++) {
        int count = 0;
        for (int j = 3; j >= 0; j--) {
            if (arr[i][j] == 1) {
                count++;
            } else {
                break;
            }
        }

        if (count > maxCount) {
            maxCount = count;
            maxRowIndex = i;
        }
    }

    return maxRowIndex;
}

int main() {
    int arr1[4][4] = {
        {0, 1, 1, 1},
        {0, 0, 1, 1},
        {1, 1, 1, 1},
        {0, 0, 0, 0}
    };

    int arr2[4][4] = {
        {0, 0, 1, 1},
        {0, 1, 1, 1},
        {0, 0, 1, 1},
        {0, 0, 0, 0}
    };

    int maxRowIndex1 = findRowWithMaxOnes(arr1, 4);
    printf("Row with maximum number of 1s in arr1: %d\n", maxRowIndex1);

    int maxRowIndex2 = findRowWithMaxOnes(arr2, 4);
    printf("Row with maximum number of 1s in arr2: %d\n", maxRowIndex2);

    return 0;
}
```

10. Given an array **arr**. Find the majority element in the array. If no majority exists, return -1. A majority element in an array is an element that appears **strictly** more than **arr.size() / 2 times** in the array.

Answer:

```
#include <stdio.h>

int findMajorityElement(int arr[], int size) {
    int count = 0;
    int candidate = -1;

    for (int i = 0; i < size; i++) {
        if (count == 0) {
            candidate = arr[i];
        }
        count += (arr[i] == candidate) ? 1 : -1;
    }

    count = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] == candidate) {
            count++;
        }
    }

    return (count > size / 2) ? candidate : -1;
}

int main() {
    int arr1[] = {1, 1, 2, 1, 3, 5, 1};
    int size1 = sizeof(arr1) / sizeof(arr1[0]);
    printf("Majority element in arr1: %d\n", findMajorityElement(arr1, size1));

    int arr2[] = {3, 3, 4, 2, 4, 4, 2, 4};
    int size2 = sizeof(arr2) / sizeof(arr2[0]);
    printf("Majority element in arr2: %d\n", findMajorityElement(arr2, size2));

    int arr3[] = {3};
    int size3 = sizeof(arr3) / sizeof(arr3[0]);
    printf("Majority element in arr3: %d\n", findMajorityElement(arr3, size3));

    return 0;
}
```

11. Given an unsorted array of integers, sort the array into a wave array. An array **arr[0..n-1]** is sorted in wave form if:
arr[0] >= arr[1] <= arr[2] >= arr[3] <= arr[4] >= …..

Answer:

```
#include <stdio.h>

int main()

{

    int array[] = {10, 49, 2, 1, 5, 23};

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

    int temp;

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

    {

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

        {

            if (array[i] > array[j])

            {
```

```
            temp = array[i];

            array[i] = array[j];

            array[j] = temp;

        }

    }

}

for (int i = 0; i < n; i = i + 2)

{

    temp = array[i];

    array[i] = array[i + 1];

    array[i + 1] = temp;

}



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

```
    {

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

    }

    return 0;

}
```