Assignment 1 Report

Assignment 1

Problem: q1.c

Question 1: Write a program to find the maximum element in an array.

Source Code

```c
/*
 * Question 1: Write a program to find the maximum element in an array.
 */

#include <stdio.h>
#include <limits.h>

int findMax(int arr[], int n) {
    if (n <= 0) {
        printf("Array is empty\n");
        return INT_MIN;
    }

    int max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    if (n <= 0) {
        printf("Array is empty\n");
        return 0;
    }

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

    int max = findMax(arr, n);
    printf("Maximum element in the array: %d\n", max);

    return 0;
}
```

Assignment 1

Execution Output

Input provided: 5 10 20 5 50 15

```
Enter the number of elements: Enter 5 elements: Maximum element in the array: 50
```

Assignment 1

Problem: q2.c

Question 2: Implement a function to reverse an array in-place.

Source Code

```
/*
 * Question 2: Implement a function to reverse an array in-place.
 */

#include <stdio.h>

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

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

        start++;
        end--;
    }
}

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

int main() {
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);

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

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

    reverseArray(arr, n);

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

    return 0;
}
```

Execution Output

Input provided: 5 1 2 3 4 5

```
Enter the number of elements: Enter 5 elements: Original array: 1 2 3 4 5
Reversed array: 5 4 3 2 1
```

Assignment 1

Problem: q3.c

Question 3: Create a program to find the intersection of two arrays.

Source Code

```c
/*
 * Question 3: Create a program to find the intersection of two arrays.
 */

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

void findIntersection(int arr1[], int n1, int arr2[], int n2) {
    printf("Intersection: ");
    int found = 0;

    // For each element in arr1, check if it exists in arr2
    for (int i = 0; i < n1; i++) {
        // Check if this element was already printed
        int alreadyPrinted = 0;
        for (int k = 0; k < i; k++) {
            if (arr1[k] == arr1[i]) {
                alreadyPrinted = 1;
                break;
            }
        }

        if (alreadyPrinted) continue;

        // Check if arr1[i] exists in arr2
        for (int j = 0; j < n2; j++) {
            if (arr1[i] == arr2[j]) {
                printf("%d ", arr1[i]);
                found = 1;
                break;
            }
        }
    }

    if (!found) {
        printf("No common elements");
    }
    printf("\n");
}

int main() {
    int n1, n2;

    printf("Enter size of first array: ");
    scanf("%d", &n1);
    int arr1[n1];
    printf("Enter %d elements for first array: ", n1);
    for (int i = 0; i < n1; i++) {
        scanf("%d", &arr1[i]);
    }

    printf("Enter size of second array: ");
    scanf("%d", &n2);
    int arr2[n2];
    printf("Enter %d elements for second array: ", n2);
    for (int i = 0; i < n2; i++) {
        scanf("%d", &arr2[i]);
    }

    findIntersection(arr1, n1, arr2, n2);

    return 0;
}
```

Execution Output

Input provided: 5 1 2 3 4 5 5 3 4 5 6 7

```
Enter size of first array: Enter 5 elements for first array: Enter size of second array: Enter 5
elements for second array: Intersection: 3 4 5
```

Assignment 1

Problem: q4.c

Question 4: Write an algorithm to rotate an array by a given number of positions.

Source Code

```c
/*
 * Question 4: Write an algorithm to rotate an array by a given number of positions.
 */

#include <stdio.h>

// Function to reverse array from start to end
void reverse(int arr[], int start, int end) {
    while (start < end) {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}

// Function to rotate array left by d positions
void rotateLeft(int arr[], int n, int d) {
    d = d % n; // Handle cases where d > n
    if (d == 0) return;

    // Reverse first d elements
    reverse(arr, 0, d - 1);
    // Reverse remaining elements
    reverse(arr, d, n - 1);
    // Reverse entire array
    reverse(arr, 0, n - 1);
}

// Function to rotate array right by d positions
void rotateRight(int arr[], int n, int d) {
    d = d % n; // Handle cases where d > n
    if (d == 0) return;

    // Reverse last d elements
    reverse(arr, n - d, n - 1);
    // Reverse first n-d elements
    reverse(arr, 0, n - d - 1);
    // Reverse entire array
    reverse(arr, 0, n - 1);
}

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

int main() {
    int n, d, direction;

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

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

    printf("Enter number of positions to rotate: ");
    scanf("%d", &d);

    printf("Enter direction (1 for left, 2 for right): ");
    scanf("%d", &direction);

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

    if (direction == 1) {
        rotateLeft(arr, n, d);
        printf("Array after left rotation by %d positions: ", d);
    } else {
        rotateRight(arr, n, d);
        printf("Array after right rotation by %d positions: ", d);
    }
    printArray(arr, n);

    return 0;
}
```

Execution Output

Input provided: 5 1 2 3 4 5 2 1

```
Enter the number of elements: Enter 5 elements: Enter number of positions to rotate: Enter
direction (1 for left, 2 for right): Original array: 1 2 3 4 5
Array after left rotation by 2 positions: 3 4 5 1 2
```

Assignment 1

Problem: q5.c

Question 5: Implement an algorithm to find the missing number in an array of integers from 1 to N.

Source Code

```c
/*
 * Question 5: Implement an algorithm to find the missing number in an array of integers from 1 to N.
 */

#include <stdio.h>

// Method 1: Using sum formula
int findMissingNumberSum(int arr[], int n) {
    // Expected sum of numbers from 1 to n+1
    long long expectedSum = (long long)(n + 1) * (n + 2) / 2;

    // Actual sum of array elements
    long long actualSum = 0;
    for (int i = 0; i < n; i++) {
        actualSum += arr[i];
    }

    return (int)(expectedSum - actualSum);
}

// Method 2: Using XOR
int findMissingNumberXOR(int arr[], int n) {
    int xor1 = 0; // XOR of all numbers from 1 to n+1
    int xor2 = 0; // XOR of all array elements

    for (int i = 1; i <= n + 1; i++) {
        xor1 ^= i;
    }

    for (int i = 0; i < n; i++) {
        xor2 ^= arr[i];
    }

    return xor1 ^ xor2;
}

int main() {
    int n;
    printf("Enter the number of elements (array should contain numbers from 1 to n+1 with one missing): ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements (one number from 1 to %d is missing): ", n, n + 1);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int missing1 = findMissingNumberSum(arr, n);
    int missing2 = findMissingNumberXOR(arr, n);

    printf("Missing number (using sum method): %d\n", missing1);
    printf("Missing number (using XOR method): %d\n", missing2);

    return 0;
}
```

Execution Output

Input provided: 4 1 2 4 5

```
Enter the number of elements (array should contain numbers from 1 to n+1 with one missing): Enter
4 elements (one number from 1 to 5 is missing): Missing number (using sum method): 3
Missing number (using XOR method): 3
```