1. Sum of Array Elements Using Pointers

Write a program that calculates the sum of elements in an integer array using pointers.

```
#include <stdio.h>

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

    int *ptr = arr;
    for (int i = 0; i < n; i++) {
        sum += *ptr;
        ptr++;
    }

    printf("Sum of array elements: %d\n", sum);
    return 0;
}
```

2. Reverse an Array Using Pointers

Write a program to reverse the elements of an integer array using pointers.

```
#include <stdio.h>

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

    int *start = arr;
    int *end = arr + n - 1;

    while (start < end) {
        int temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }

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

```
    return 0;
}
```

3. Find the Maximum Element in an Array Using Pointers

Write a program to find the maximum element in an integer array using pointers.

```
#include <stdio.h>

int main() {
    int arr[] = {12, 45, 23, 67, 9, 51};
    int n = sizeof(arr) / sizeof(arr[0]);

    int *ptr = arr;
    int max = *ptr;

    for (int i = 1; i < n; i++) {
        if (*(ptr + i) > max) {
            max = *(ptr + i);
        }
    }

    printf("Maximum element in the array: %d\n", max);
    return 0;
}
```

4. Copy One Array to Another Using Pointers

Write a program to copy the elements from one integer array to another using pointers.

```
#include <stdio.h>

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

    int *src_ptr = source;
    int *dest_ptr = destination;

    for (int i = 0; i < n; i++) {
        *dest_ptr = *src_ptr;
        src_ptr++;
        dest_ptr++;
    }

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

    printf("\nCopied array: ");
    for (int i = 0; i < n; i++) {
```

```
        printf("%d ", destination[i]);
    }

    printf("\n");

    return 0;
}
```

5. Search for an Element in an Array Using Pointers

Write a program to search for a specific element in an integer array using pointers.

```
#include <stdio.h>

int main() {
    int arr[] = {2, 4, 6, 8, 10};
    int n = sizeof(arr) / sizeof(arr[0]);
    int searchValue = 6;

    int *ptr = arr;
    int found = 0;

    for (int i = 0; i < n; i++) {
        if (*ptr == searchValue) {
            found = 1;
            break;
        }
```

```
        ptr++;
    }

    if (found) {
        printf("Element %d found in the array.\n", searchValue);
    } else {
        printf("Element %d not found in the array.\n", searchValue);
    }

    return 0;
}
```

6. Calculate the Average of Array Elements Using Pointers

Write a program to calculate the average of elements in an integer array using pointers.

```
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int n = sizeof(arr) / sizeof(arr[0]);

    int *ptr = arr;
    int sum = 0;

    for (int i = 0; i < n; i++) {
        sum += *ptr;
```

```
        ptr++;
    }

    float average = (float)sum / n;
    printf("Average of array elements: %.2f\n", average);
    return 0;
}
```

7. Merge Two Arrays Using Pointers

Write a program to merge two integer arrays into a third array using pointers.

```
#include <stdio.h>

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

    int *ptr1 = arr1;
    int *ptr2 = arr2;
    int *merged_ptr = merged;

    for (int i = 0; i < n1; i++) {
        *merged_ptr = *ptr1;
```

```
        ptr1++;
        merged_ptr++;
    }

    for (int i = 0; i < n2; i++) {
        *merged_ptr = *ptr2;
        ptr2++;
        merged_ptr++;
    }

    printf("Merged array: ");
    for (int i = 0; i < n1 + n2; i++) {
        printf("%d ", merged[i]);
    }
    printf("\n");

    return 0;
}
```

8. Remove Duplicates from an Array Using Pointers

Write a program to remove duplicate elements from an integer array using pointers.

```
#include <stdio.h>

int main() {
    int arr[] = {2, 3, 4, 2, 7, 4, 8, 7};
    int n = sizeof(arr) / sizeof(arr[0]);

    int *ptr = arr;

    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; ) {
            if (*(ptr + i) == *(ptr + j)) {
                for (int k = j; k < n - 1; k++) {
                    *(ptr + k) = *(ptr + k + 1);
                }
                n--;
            } else {
                j++;
            }
        }
    }

    printf("Array with duplicates removed: ");
    for (int i = 0; i < n; i++) {
```

```
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

9. Find the Intersection of Two Arrays Using Pointers

Write a program to find the intersection of two integer arrays using pointers.

```
#include <stdio.h>

int main() {
    int arr1[] = {2, 3, 4, 5, 6};
    int arr2[] = {4, 5, 6, 7, 8};
    int n1 = sizeof(arr1) / sizeof(arr1[0]);
    int n2 = sizeof(arr2) / sizeof(arr2[0]);

    printf("Intersection of arrays: ");
    for (int i = 0; i < n1; i++) {
        for (int j = 0; j < n2; j++) {
            if (arr1[i] == arr2[j]) {
                printf("%d ", arr1[i]);
                break;
            }
        }
    }
    printf("\n");

    return 0;
}
```

10. Split an Array into Two Arrays Using Pointers

Write a program to split an integer array into two arrays based on a given split point using pointers.

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 3, 4, 5, 6, 7, 8};
    int n = sizeof(arr) / sizeof(arr[0]);
    int splitIndex = 4;
    int arr1[splitIndex];
    int arr2[n - splitIndex];

    int *ptr = arr;
    int *ptr1 = arr1;
    int *ptr2 = arr2;

    for (int i = 0; i < n; i++) {
        if (i < splitIndex) {
            *ptr1 = *ptr;
            ptr1++;
        } else {
            *ptr2 = *ptr;
            ptr2++;
        }
        ptr++;
    }
```

```
    printf("Array 1: ");
    for (int i = 0; i < splitIndex; i++) {
        printf("%d ", arr1[i]);
    }

    printf("\nArray 2: ");
    for (int i = 0; i < n - splitIndex; i++) {
        printf("%d ", arr2[i]);
    }
    printf("\n");

    return 0;
}
```