1. Swap two numbers using pointers

Input-Output Format

makefile
CopyEdit

```
Input:
a = 5, b = 10

Output:
a = 10, b = 5
```

Solution

c
CopyEdit

```c
#include <stdio.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int a = 5, b = 10;
    swap(&a, &b);
    printf("a = %d, b = %d\n", a, b);
    return 0;
}
```

---

2. Find the length of a string using a pointer

Input-Output Format

makefile
CopyEdit

```
Input:
Hello

Output:
Length = 5
```

Solution

c
CopyEdit

```
#include <stdio.h>

int stringLength(char *str) {
    int length = 0;
    while (*str != '\0') {
        length++;
        str++;
    }
    return length;
}

int main() {
    char str[] = "Hello";
    printf("Length = %d\n", stringLength(str));
    return 0;
}
```

---

3. Reverse a string using pointers

Input-Output Format

makefile
CopyEdit

```
Input:
Hello
```

```
Output:
olleH
```

Solution

c
CopyEdit

```
#include <stdio.h>

void reverseString(char *str) {
    char *start = str;
    char *end = str;
    while (*end != '\0') {
        end++;
    }
    end--;

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

int main() {
    char str[] = "Hello";
    reverseString(str);
    printf("%s\n", str);
    return 0;
}
```

---

4. Compare two strings using pointers

Input-Output Format

makefile
CopyEdit

```
Input:
str1 = "Hello", str2 = "Hello"

Output:
Strings are equal.
```

Solution

c
CopyEdit

```c
#include <stdio.h>

int compareStrings(char *str1, char *str2) {
    while (*str1 != '\0' && *str2 != '\0') {
        if (*str1 != *str2) {
            return 0;
        }
        str1++;
        str2++;
    }
    return (*str1 == '\0' && *str2 == '\0');
}

int main() {
    char str1[] = "Hello", str2[] = "Hello";
    if (compareStrings(str1, str2)) {
        printf("Strings are equal.\n");
    } else {
        printf("Strings are not equal.\n");
    }
    return 0;
}
```

---

5. Find the largest element in an array using pointers

Input-Output Format

makefile
CopyEdit

```
Input:
arr[] = {1, 4, 6, 2, 9}

Output:
Largest = 9
```

Solution

c
CopyEdit

```c
#include <stdio.h>

int findLargest(int *arr, int size) {
    int max = *arr;
    for (int i = 1; i < size; i++) {
        if (*(arr + i) > max) {
            max = *(arr + i);
        }
    }
    return max;
}

int main() {
    int arr[] = {1, 4, 6, 2, 9};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Largest = %d\n", findLargest(arr, size));
    return 0;
}
```

---

6. Add two matrices using pointers

Input-Output Format

yaml
CopyEdit

```
Input:
Matrix 1:
1 2
3 4

Matrix 2:
5 6
7 8

Output:
Matrix after addition:
6 8
10 12
```

Solution

c
CopyEdit

```c
#include <stdio.h>

void addMatrices(int (*mat1)[2], int (*mat2)[2], int
(*result)[2]) {
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}

int main() {
    int mat1[2][2] = {{1, 2}, {3, 4}};
    int mat2[2][2] = {{5, 6}, {7, 8}};
```

```
    int result[2][2];

    addMatrices(mat1, mat2, result);

    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            printf("%d ", result[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

---

7. Count the occurrences of a character in a string using pointers

Input-Output Format

vbnet
CopyEdit

```
Input:
String: "hello world"
Character: 'o'

Output:
Occurrences = 2
```

Solution

c
CopyEdit

```
#include <stdio.h>

int countOccurrences(char *str, char ch) {
    int count = 0;
    while (*str != '\0') {
        if (*str == ch) {
```

```
            count++;
        }
        str++;
    }
    return count;
}

int main() {
    char str[] = "hello world";
    char ch = 'o';
    printf("Occurrences = %d\n", countOccurrences(str, ch));
    return 0;
}
```

---

8. Print an array using pointers

Input-Output Format

css
CopyEdit

```
Input:
arr[] = {1, 2, 3, 4}

Output:
1 2 3 4
```

Solution

c
CopyEdit

```
#include <stdio.h>

void printArray(int *arr, int size) {
    for (int i = 0; i < size; i++) {
        printf("%d ", *(arr + i));
    }
```

```
    printf("\n");
}

int main() {
    int arr[] = {1, 2, 3, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    printArray(arr, size);
    return 0;
}
```

---

9. Find the sum of an array using pointers

Input-Output Format

makefile
CopyEdit

```
Input:
arr[] = {1, 2, 3, 4}

Output:
Sum = 10
```

Solution

c
CopyEdit

```
#include <stdio.h>

int sumArray(int *arr, int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += *(arr + i);
    }
    return sum;
}
```

```
int main() {
    int arr[] = {1, 2, 3, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Sum = %d\n", sumArray(arr, size));
    return 0;
}
```

---

10. Copy one array to another using pointers

Input-Output Format

yaml
CopyEdit

```
Input:
arr[] = {1, 2, 3, 4}

Output:
Copied Array: 1 2 3 4
```

Solution

c
CopyEdit

```
#include <stdio.h>

void copyArray(int *src, int *dest, int size) {
    for (int i = 0; i < size; i++) {
        *(dest + i) = *(src + i);
    }
}

int main() {
    int arr[] = {1, 2, 3, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    int copiedArr[size];
```

```
    copyArray(arr, copiedArr, size);

    printf("Copied Array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", copiedArr[i]);
    }
    printf("\n");
    return 0;
}
```

---

11. Find the first occurrence of an element in an array using pointers

Input-Output Format

makefile
CopyEdit

```
Input:
arr[] = {1, 2, 3, 4}
Element: 3

Output:
First occurrence at index 2
```

Solution

c
CopyEdit

```
#include <stdio.h>

int findFirstOccurrence(int *arr, int size, int element) {
    for (int i = 0; i < size; i++) {
        if (*(arr + i) == element) {
            return i;
        }
    }
    return -1;
```

```
}

int main() {
    int arr[] = {1, 2, 3, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    int element = 3;
    int index = findFirstOccurrence(arr, size, element);

    if (index != -1) {
        printf("First occurrence at index %d\n", index);
    } else {
        printf("Element not found\n");
    }

    return 0;
}
```

---

12. Sort an array in ascending order using pointers

Input-Output Format

yaml
CopyEdit

```
Input:
arr[] = {5, 2, 8, 3}

Output:
Sorted Array: 2 3 5 8
```

Solution

c
CopyEdit

```
#include <stdio.h>

void sortArray(int *arr, int size) {
```

```
    for (int i = 0; i < size - 1; i++) {
        for (int j = i + 1; j < size; j++) {
            if (*(arr + i) > *(arr + j)) {
                int temp = *(arr + i);
                *(arr + i) = *(arr + j);
                *(arr + j) = temp;
            }
        }
    }
}

int main() {
    int arr[] = {5, 2, 8, 3};
    int size = sizeof(arr) / sizeof(arr[0]);

    sortArray(arr, size);

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

    return 0;
}
```

---

13. Find the second largest element in an array using pointers

Input-Output Format

css
CopyEdit
`Input:`
`arr[] = {10, 5, 2, 8, 1}`

```
Output:
Second largest = 8
```

Solution

c
CopyEdit

```
#include <stdio.h>

int findSecondLargest(int *arr, int size) {
    int largest = *arr, secondLargest = *arr;
    for (int i = 1; i < size; i++) {
        if (*(arr + i) > largest) {
            secondLargest = largest;
            largest = *(arr + i);
        } else if (*(arr + i) > secondLargest && *(arr + i) !=
largest) {
            secondLargest = *(arr + i);
        }
    }
    return secondLargest;
}

int main() {
    int arr[] = {10, 5, 2, 8, 1};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Second largest = %d\n", findSecondLargest(arr,
size));
    return 0;
}
```

---

14. Check if an array is sorted using pointers

Input-Output Format

css

CopyEdit

```
Input:
arr[] = {1, 2, 3, 4, 5}

Output:
Array is sorted
```

Solution

c

CopyEdit

```c
#include <stdio.h>

int isSorted(int *arr, int size) {
    for (int i = 1; i < size; i++) {
        if (*(arr + i) < *(arr + i - 1)) {
            return 0;  // Not sorted
        }
    }
    return 1;  // Sorted
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    if (isSorted(arr, size)) {
        printf("Array is sorted\n");
    } else {
        printf("Array is not sorted\n");
    }
    return 0;
}
```

---

15. Find the minimum and maximum element in an array using pointers

Input-Output Format

makefile
CopyEdit

```
Input:
arr[] = {5, 1, 9, 2}

Output:
Min = 1, Max = 9
```

Solution

c
CopyEdit

```c
#include <stdio.h>

void findMinMax(int *arr, int size, int *min, int *max) {
    *min = *arr;
    *max = *arr;
    for (int i = 1; i < size; i++) {
        if (*(arr + i) < *min) {
            *min = *(arr + i);
        }
        if (*(arr + i) > *max) {
            *max = *(arr + i);
        }
    }
}

int main() {
    int arr[] = {5, 1, 9, 2};
    int size = sizeof(arr) / sizeof(arr[0]);
    int min, max;
    findMinMax(arr, size, &min, &max);
    printf("Min = %d, Max = %d\n", min, max);
    return 0;
}
```

---

16. Find the sum of diagonals of a 2D array using pointers

Input-Output Format

makefile
CopyEdit

```
Input:
Matrix:
1 2 3
4 5 6
7 8 9

Output:
Sum of diagonals = 25
```

Solution

c
CopyEdit

```c
#include <stdio.h>

int sumDiagonals(int (*arr)[3], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i][i];  // Main diagonal
        sum += arr[i][size - i - 1];  // Secondary diagonal
    }
    return sum;
}

int main() {
    int arr[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    printf("Sum of diagonals = %d\n", sumDiagonals(arr, 3));
    return 0;
}
```

17. Check if an element exists in an array using pointers

Input-Output Format:

css
CopyEdit

```
Input: {1, 2, 3, 4, 5}, 3
Output: Element found
```

Solution:

c
CopyEdit

```c
#include <stdio.h>

void findElement(int *arr, int n, int x) {
    for (int i = 0; i < n; i++) {
        if (*(arr + i) == x) {
            printf("Element found\n");
            return;
        }
    }
    printf("Element not found\n");
}

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

---

18. Add two matrices using pointers

Input-Output Format:

java

CopyEdit

```
Input:
Matrix A = {{1, 2}, {3, 4}}
Matrix B = {{5, 6}, {7, 8}}

Output:
Matrix C = {{6, 8}, {10, 12}}
```

Solution:

c

CopyEdit

```
#include <stdio.h>

void addMatrices(int *a, int *b, int *c, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            *(c + i * cols + j) = *(a + i * cols + j) + *(b + i
* cols + j);
        }
    }
}

int main() {
    int A[2][2] = {{1, 2}, {3, 4}};
    int B[2][2] = {{5, 6}, {7, 8}};
    int C[2][2];
    int rows = 2, cols = 2;

    addMatrices((int *)A, (int *)B, (int *)C, rows, cols);

    printf("Resultant Matrix C:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", C[i][j]);
        }
```

```
        printf("\n");
    }
    return 0;
}
```

---

19. Multiply two matrices using pointers

Input-Output Format:

java
CopyEdit

```
Input:
Matrix A = {{1, 2}, {3, 4}}
Matrix B = {{5, 6}, {7, 8}}

Output:
Matrix C = {{19, 22}, {43, 50}}
```

Solution:

c
CopyEdit

```
#include <stdio.h>

void multiplyMatrices(int *a, int *b, int *c, int rowsA, int
colsA, int rowsB, int colsB) {
    for (int i = 0; i < rowsA; i++) {
        for (int j = 0; j < colsB; j++) {
            *(c + i * colsB + j) = 0;
            for (int k = 0; k < colsA; k++) {
                *(c + i * colsB + j) += (*(a + i * colsA + k)) *
(*(b + k * colsB + j));
            }
        }
    }
}
```

```
int main() {
    int A[2][2] = {{1, 2}, {3, 4}};
    int B[2][2] = {{5, 6}, {7, 8}};
    int C[2][2];

    multiplyMatrices((int *)A, (int *)B, (int *)C, 2, 2, 2, 2);

    printf("Resultant Matrix C:\n");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

---

20. Transpose a matrix using pointers

Input-Output Format:

css
CopyEdit

```
Input:
Matrix A = {{1, 2}, {3, 4}}

Output:
Matrix A = {{1, 3}, {2, 4}}
```

Solution:

c
CopyEdit

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

```
void transposeMatrix(int *a, int *b, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            *(b + j * rows + i) = *(a + i * cols + j);
        }
    }
}

int main() {
    int A[2][2] = {{1, 2}, {3, 4}};
    int B[2][2];
    int rows = 2, cols = 2;

    transposeMatrix((int *)A, (int *)B, rows, cols);

    printf("Transposed Matrix B:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", B[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

---

21. Find the diagonal sum of a matrix using pointers

Input-Output Format:

css
CopyEdit

```
Input:
Matrix A = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}

Output: 15
```

Solution:

c
CopyEdit

```
#include <stdio.h>

int diagonalSum(int *a, int rows, int cols) {
    int sum = 0;
    for (int i = 0; i < rows && i < cols; i++) {
        sum += *(a + i * cols + i);
    }
    return sum;
}

int main() {
    int A[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int rows = 3, cols = 3;
    printf("Diagonal sum: %d\n", diagonalSum((int *)A, rows,
cols));
    return 0;
}
```

---

22. Check if a matrix is identity matrix using pointers

Input-Output Format:

yaml
CopyEdit

```
Input:
Matrix A = {{1, 0}, {0, 1}}

Output: Yes
```

Solution:

c
CopyEdit

```c
#include <stdio.h>

int isIdentity(int *a, int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (i == j && *(a + i * cols + j) != 1) {
                return 0;
            } else if (i != j && *(a + i * cols + j) != 0) {
                return 0;
            }
        }
    }
    return 1;
}

int main() {
    int A[2][2] = {{1, 0}, {0, 1}};
    int rows = 2, cols = 2;
    if (isIdentity((int *)A, rows, cols)) {
        printf("Yes\n");
    } else {
        printf("No\n");
    }
    return 0;
}
```

---

23. Add two numbers without using the '+' operator using pointers

Input-Output Format:

makefile
CopyEdit
`Input: 5, 3`
`Output: 8`

Solution:

c
CopyEdit

```
#include <stdio.h>

int add(int *a, int *b) {
    while (*b != 0) {
        *a = *a + 1;
        *b = *b - 1;
    }
    return *a;
}

int main() {
    int x = 5, y = 3;
    printf("Sum: %d\n", add(&x, &y));
    return 0;
}
```

24. Find the address of the first element of an array

Input-Output Format:

css
CopyEdit

```
Input: {1, 2, 3, 4, 5}
Output: Address of first element
```

Solution:

c
CopyEdit

```
#include <stdio.h>

void printAddress(int *arr) {
    printf("Address of the first element: %p\n", (void *)arr);
}
```

```
int main() {
    int arr[] = {1, 2, 3, 4, 5};
    printAddress(arr);
    return 0;
}
```

---

25. Find the sum of all elements in an array using pointers

Input-Output Format:

css
CopyEdit

```
Input: {1, 2, 3, 4, 5}
Output: Sum = 15
```

Solution:

c
CopyEdit

```
#include <stdio.h>

int sumArray(int *arr, int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += *(arr + i);
    }
    return sum;
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Sum of array elements: %d\n", sumArray(arr, size));
    return 0;
}
```

---

26. Reverse an array using pointers

Input-Output Format:

css
CopyEdit

```
Input: {1, 2, 3, 4, 5}
Output: {5, 4, 3, 2, 1}
```

Solution:

c
CopyEdit

```c
#include <stdio.h>

void reverseArray(int *arr, int size) {
    int *start = arr;
    int *end = arr + size - 1;
    while (start < end) {
        int temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    reverseArray(arr, size);
    printf("Reversed array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
```

```
    printf("\n");
    return 0;
}
```

---

27. Find the largest element in an array using pointers

Input-Output Format:

css
CopyEdit

```
Input: {1, 2, 3, 4, 5}
Output: Largest element = 5
```

Solution:

c
CopyEdit

```
#include <stdio.h>

int findLargest(int *arr, int size) {
    int *largest = arr;
    for (int i = 1; i < size; i++) {
        if (*(arr + i) > *largest) {
            largest = arr + i;
        }
    }
    return *largest;
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Largest element: %d\n", findLargest(arr, size));
    return 0;
}
```

---

28. Find the second largest element in an array using pointers

Input-Output Format:

css
CopyEdit

```
Input: {1, 2, 3, 4, 5}
Output: Second largest element = 4
```

Solution:

c
CopyEdit

```
#include <stdio.h>

int findSecondLargest(int *arr, int size) {
    int *largest = arr, *secondLargest = arr;
    for (int i = 1; i < size; i++) {
        if (*(arr + i) > *largest) {
            secondLargest = largest;
            largest = arr + i;
        } else if (*(arr + i) > *secondLargest && *(arr + i) !=
*largest) {
            secondLargest = arr + i;
        }
    }
    return *secondLargest;
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Second largest element: %d\n",
findSecondLargest(arr, size));
    return 0;
}
```

---

29. Check if a number is present in an array using pointers

Input-Output Format:

css
CopyEdit

```
Input: {1, 2, 3, 4, 5}, 3
Output: Element found
```

Solution:

c
CopyEdit

```c
#include <stdio.h>

void findElement(int *arr, int size, int x) {
    for (int i = 0; i < size; i++) {
        if (*(arr + i) == x) {
            printf("Element found\n");
            return;
        }
    }
    printf("Element not found\n");
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    int x = 3;
    findElement(arr, size, x);
    return 0;
}
```

---

30. Swap two elements in an array using pointers

Input-Output Format:

css
CopyEdit

```
Input: {1, 2, 3, 4}, Swap index 1 and 3
Output: {1, 4, 3, 2}
```

Solution:

c
CopyEdit

```c
#include <stdio.h>

void swapElements(int *arr, int i, int j) {
    int temp = *(arr + i);
    *(arr + i) = *(arr + j);
    *(arr + j) = temp;
}

int main() {
    int arr[] = {1, 2, 3, 4};
    int i = 1, j = 3;
    swapElements(arr, i, j);
    printf("Array after swap: ");
    for (int k = 0; k < 4; k++) {
        printf("%d ", arr[k]);
    }
    printf("\n");
    return 0;
}
```

---

31. Count the occurrences of a specific element in an array using pointers

Input-Output Format:

css
CopyEdit

```
Input: {1, 2, 3, 2, 4}, 2
Output: Occurrences of 2 = 2
```

Solution:

c
CopyEdit

```
#include <stdio.h>

int countOccurrences(int *arr, int size, int x) {
    int count = 0;
    for (int i = 0; i < size; i++) {
        if (*(arr + i) == x) {
            count++;
        }
    }
    return count;
}

int main() {
    int arr[] = {1, 2, 3, 2, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    int x = 2;
    printf("Occurrences of %d: %d\n", x, countOccurrences(arr,
size, x));
    return 0;
}
```

---

32. Copy elements from one array to another using pointers

Input-Output Format:

makefile
CopyEdit

```
Input: {1, 2, 3, 4}, Target array size = 4
Output: {1, 2, 3, 4}
```

Solution:

c
CopyEdit

```
#include <stdio.h>

void copyArray(int *source, int *target, int size) {
    for (int i = 0; i < size; i++) {
        *(target + i) = *(source + i);
    }
}

int main() {
    int source[] = {1, 2, 3, 4};
    int target[4];
    int size = sizeof(source) / sizeof(source[0]);

    copyArray(source, target, size);

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

---

33. Print elements of an array using pointers

Input-Output Format:

css
CopyEdit

```
Input: {1, 2, 3, 4, 5}
Output: 1 2 3 4 5
```

Solution:

c
CopyEdit

```
#include <stdio.h>

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

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

---

34. Find the minimum element in an array using pointers

Input-Output Format:

css
CopyEdit

```
Input: {4, 3, 7, 2, 5}
Output: Minimum element = 2
```

Solution:

c
CopyEdit

```
#include <stdio.h>

int findMinimum(int *arr, int size) {
```

```
    int *min = arr;
    for (int i = 1; i < size; i++) {
        if (*(arr + i) < *min) {
            min = arr + i;
        }
    }
    return *min;
}

int main() {
    int arr[] = {4, 3, 7, 2, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Minimum element: %d\n", findMinimum(arr, size));
    return 0;
}
```

---

35. Sum of squares of array elements using pointers

Input-Output Format:

css
CopyEdit
```
Input: {1, 2, 3, 4}
Output: 30
```

Solution:

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

int sumOfSquares(int *arr, int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += *(arr + i) * *(arr + i);
    }
```

```
    return sum;
}

int main() {
    int arr[] = {1, 2, 3, 4};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Sum of squares: %d\n", sumOfSquares(arr, size));
    return 0;
}
```

---

36. Find the average of the elements in an array using pointers

Input-Output Format:

css
CopyEdit

```
Input: {1, 2, 3, 4, 5}
Output: Average = 3
```

Solution:

c
CopyEdit

```
#include <stdio.h>

float findAverage(int *arr, int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += *(arr + i);
    }
    return (float)sum / size;
}

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

```
    printf("Average: %.2f\n", findAverage(arr, size));
    return 0;
}
```

37. Merge two sorted arrays into one sorted array using pointers (continued)

c
CopyEdit

```
            *(merged + k++) = *(b + j++);
        }
    }
    while (i < sizeA) {
        *(merged + k++) = *(a + i++);
    }
    while (j < sizeB) {
        *(merged + k++) = *(b + j++);
    }
}

int main() {
    int A[] = {1, 3, 5};
    int B[] = {2, 4, 6};
    int sizeA = sizeof(A) / sizeof(A[0]);
    int sizeB = sizeof(B) / sizeof(B[0]);
    int merged[sizeA + sizeB];

    mergeArrays(A, B, merged, sizeA, sizeB);

    printf("Merged Array: ");
    for (int i = 0; i < sizeA + sizeB; i++) {
        printf("%d ", merged[i]);
    }
    printf("\n");
    return 0;
}
```

---

38. Rotate an array to the right by k positions using pointers

Input-Output Format:

makefile
CopyEdit

```
Input: {1, 2, 3, 4, 5}, k = 2
Output: {4, 5, 1, 2, 3}
```

Solution:

c
CopyEdit

```
#include <stdio.h>

void rotateArray(int *arr, int size, int k) {
    k = k % size;  // In case k is greater than the size of the
array
    int temp[size];

    for (int i = 0; i < size; i++) {
        temp[(i + k) % size] = *(arr + i);
    }

    for (int i = 0; i < size; i++) {
        *(arr + i) = temp[i];
    }
}

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

    rotateArray(arr, size, k);

    printf("Array after rotation: ");
    for (int i = 0; i < size; i++) {
```

```
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

---

39. Check if an array is sorted in ascending order using pointers

Input-Output Format:

pgsql
CopyEdit

```
Input: {1, 2, 3, 4, 5}
Output: Array is sorted
```

Solution:

c
CopyEdit

```
#include <stdio.h>

int isSorted(int *arr, int size) {
    for (int i = 0; i < size - 1; i++) {
        if (*(arr + i) > *(arr + i + 1)) {
            return 0;  // Array is not sorted
        }
    }
    return 1;  // Array is sorted
}

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

    if (isSorted(arr, size)) {
        printf("Array is sorted\n");
```

```
    } else {
        printf("Array is not sorted\n");
    }

    return 0;
}
```

---

40. Find the length of a string using pointers

Input-Output Format:

vbnet
CopyEdit
`Input: "Hello"`
`Output: Length = 5`

Solution:

c
CopyEdit

```
#include <stdio.h>

int stringLength(char *str) {
    int length = 0;
    while (*str != '\0') {
        length++;
        str++;
    }
    return length;
}

int main() {
    char str[] = "Hello";
    printf("Length of the string: %d\n", stringLength(str));
    return 0;
}
```

---

41. Concatenate two strings using pointers

Input-Output Format:

vbnet
CopyEdit

```
Input: "Hello", " World"
Output: "Hello World"
```

Solution:

c
CopyEdit

```c
#include <stdio.h>

void concatenateStrings(char *str1, char *str2) {
    while (*str1 != '\0') {
        str1++;
    }
    while (*str2 != '\0') {
        *str1 = *str2;
        str1++;
        str2++;
    }
    *str1 = '\0';
}

int main() {
    char str1[50] = "Hello";
    char str2[] = " World";

    concatenateStrings(str1, str2);

    printf("Concatenated string: %s\n", str1);
    return 0;
```

```
}
```

---

42. Compare two strings using pointers

Input-Output Format:

vbnet
CopyEdit

```
Input: "Hello", "Hello"
Output: Strings are equal
```

Solution:

c
CopyEdit

```
#include <stdio.h>

int compareStrings(char *str1, char *str2) {
    while (*str1 != '\0' && *str2 != '\0') {
        if (*str1 != *str2) {
            return 0;  // Strings are not equal
        }
        str1++;
        str2++;
    }
    return (*str1 == *str2);  // Check if both strings are at
their end
}

int main() {
    char str1[] = "Hello";
    char str2[] = "Hello";

    if (compareStrings(str1, str2)) {
        printf("Strings are equal\n");
    } else {
```

```
        printf("Strings are not equal\n");
    }

    return 0;
}
```

---

43. Copy one string to another using pointers

Input-Output Format:

vbnet
CopyEdit

```
Input: "Hello"
Output: "Hello"
```

Solution:

c
CopyEdit

```
#include <stdio.h>

void copyString(char *src, char *dest) {
    while (*src != '\0') {
        *dest = *src;
        src++;
        dest++;
    }
    *dest = '\0';
}

int main() {
    char src[] = "Hello";
    char dest[50];

    copyString(src, dest);
```

```
    printf("Copied string: %s\n", dest);
    return 0;
}
```

---

44. Count the number of vowels in a string using pointers

Input-Output Format:

vbnet
CopyEdit

```
Input: "Hello"
Output: Number of vowels = 2
```

Solution:

c
CopyEdit

```
#include <stdio.h>

int countVowels(char *str) {
    int count = 0;
    while (*str != '\0') {
        if (*str == 'a' || *str == 'e' || *str == 'i' || *str ==
'o' || *str == 'u' ||
            *str == 'A' || *str == 'E' || *str == 'I' || *str ==
'O' || *str == 'U') {
            count++;
        }
        str++;
    }
    return count;
}

int main() {
    char str[] = "Hello";
    printf("Number of vowels: %d\n", countVowels(str));
```

```
    return 0;
}
```

---

45. Convert a string to uppercase using pointers

Input-Output Format:

vbnet
CopyEdit
`Input: "Hello"`
`Output: "HELLO"`

Solution:

c
CopyEdit

```
#include <stdio.h>

void toUpperCase(char *str) {
    while (*str != '\0') {
        if (*str >= 'a' && *str <= 'z') {
            *str = *str - 'a' + 'A';
        }
        str++;
    }
}

int main() {
    char str[] = "Hello";
    toUpperCase(str);
    printf("Uppercase string: %s\n", str);
    return 0;
}
```

---

46. Convert a string to lowercase using pointers

Input-Output Format:

vbnet
CopyEdit

```
Input: "HELLO"
Output: "hello"
```

Solution:

c
CopyEdit

```c
#include <stdio.h>

void toLowerCase(char *str) {
    while (*str != '\0') {
        if (*str >= 'A' && *str <= 'Z') {
            *str = *str + 'a' - 'A';
        }
        str++;
    }
}

int main() {
    char str[] = "HELLO";
    toLowerCase(str);
    printf("Lowercase string: %s\n", str);
    return 0;
}
```

---

47. Reverse a string using pointers

Input-Output Format:

vbnet
CopyEdit

```
Input: "Hello"
Output: "olleH"
```

Solution:

c
CopyEdit

```
#include <stdio.h>

void reverseString(char *str) {
    char *end = str;
    while (*end != '\0') {
        end++;
    }
    end--;  // Point to the last character

    while (str < end) {
        char temp = *str;
        *str = *end;
        *end = temp;
        str++;
        end--;
    }
}

int main() {
    char str[] = "Hello";
    reverseString(str);
    printf("Reversed string: %s\n", str);
    return 0;
}
```

---

48. Find the first occurrence of a character in a string using pointers

Input-Output Format:

vbnet
CopyEdit

```
Input: "Hello", 'e'
Output: First occurrence at index 1
```

Solution:

c

CopyEdit

```
#include <stdio.h>

int findFirstOccurrence(char *str, char ch) {
    int index = 0;
    while (*str != '\0') {
        if (*str == ch) {
            return index;
        }
        str++;
        index++;
    }
    return -1;  // Character not found
}

int main() {
    char str[] = "Hello";
    char ch = 'e';
    int index = findFirstOccurrence(str, ch);

    if (index != -1) {
        printf("First occurrence of '%c' is at index %d\n", ch,
index);
    } else {
        printf("Character not found\n");
    }

    return 0;
}
```