

1. Given a sorted array of integers, write an efficient algorithm to find a pair of elements that sum up to a given target value. 2. Given a string, write a function to reverse the order of words in the string. For example, "the sky is blue" becomes "blue is sky the". 3. Write a function to remove all occurrences of a given character from a string. 4. Generate all possible permutations of a given string. 5. Determine whether a given string is a palindrome.

ChatGPT said:

ChatGPT

Here are the solutions for the given problems:

1. Find a Pair of Elements that Sum Up to a Given Target Value (Sorted Array)

Pseudocode:

c

Copy code

```
// Initialize array and target sum
int arr[] = {1, 2, 4, 5, 6, 8, 9};
int n = 7; // size of the array
int target = 10;

// Use two-pointer technique
int left = 0, right = n - 1;
while (left < right) {
    int current_sum = arr[left] + arr[right];
    if (current_sum == target) {
        // Found the pair
        print(arr[left], arr[right]);
        break;
    } else if (current_sum < target) {
        left++; // Move left pointer to the right
    } else {
        right--; // Move right pointer to the left
    }
}
```

Code:

c

Copy code

```
#include <stdio.h>

int main() {
    int arr[] = {1, 2, 4, 5, 6, 8, 9};
    int n = 7, target = 10;
    int left = 0, right = n - 1;

    while (left < right) {
        int current_sum = arr[left] + arr[right];
        if (current_sum == target) {
            printf("Pair found: %d, %d\n", arr[left], arr[right]);
            break;
        } else if (current_sum < target) {
            left++;
        } else {
            right--;
        }
    }

    return 0;
}
```

2. Reverse the Order of Words in a String

Pseudocode:

c

Copy code

```
// Initialize the string
char str[] = "the sky is blue";
int n = strlen(str);

// Step 1: Reverse the entire string
reverse(str, 0, n - 1);

// Step 2: Reverse each word in the reversed string
```

```

int start = 0;
for (int i = 0; i <= n; i++) {
    if (str[i] == ' ' || str[i] == '\0') {
        reverse(str, start, i - 1);
        start = i + 1;
    }
}

```

Code:

c

Copy code

```

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

```

```

// Function to reverse a portion of a string
void reverse(char str[], int start, int end) {
    while (start < end) {
        char temp = str[start];
        str[start] = str[end];
        str[end] = temp;
        start++;
        end--;
    }
}

```

```

int main() {
    char str[] = "the sky is blue";
    int n = strlen(str);

    // Reverse the entire string
    reverse(str, 0, n - 1);

    // Reverse each word
    int start = 0;
    for (int i = 0; i <= n; i++) {
        if (str[i] == ' ' || str[i] == '\0') {
            reverse(str, start, i - 1);
            start = i + 1;
        }
    }
}

```

```

        }
    }

    printf("Reversed string: %s\n", str);
    return 0;
}

```

3. Remove All Occurrences of a Given Character from a String

Pseudocode:

```

c
Copy code
// Initialize string and character to remove
char str[] = "hello world";
char remove_char = 'o';

// Initialize a new string to store the result
int j = 0;
for (int i = 0; str[i] != '\0'; i++) {
    if (str[i] != remove_char) {
        str[j] = str[i]; // Copy the character to new string
        j++;
    }
}
str[j] = '\0'; // Null-terminate the new string

```

Code:

```

c
Copy code
#include <stdio.h>

int main() {
    char str[] = "hello world";
    char remove_char = 'o';
    int j = 0;

```

```

    for (int i = 0; str[i] != '\0'; i++) {
        if (str[i] != remove_char) {
            str[j] = str[i];
            j++;
        }
    }
    str[j] = '\0'; // Null-terminate the string

    printf("Modified string: %s\n", str);
    return 0;
}

```

4. Generate All Possible Permutations of a Given String

Pseudocode:

```

c
Copy code
// Function to swap characters in the string
swap(char str[], int i, int j);

// Function to generate permutations using recursion
permutations(char str[], int l, int r) {
    if (l == r) {
        print(str);
    } else {
        for (int i = l; i <= r; i++) {
            swap(str, l, i); // Swap the current index with the first
            permutations(str, l + 1, r); // Generate permutations for
the rest
            swap(str, l, i); // Swap back to restore original order
        }
    }
}

```

Code:

c

Copy code

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

// Function to swap characters in a string
void swap(char str[], int i, int j) {
    char temp = str[i];
    str[i] = str[j];
    str[j] = temp;
}

// Function to generate all permutations of the string
void permutations(char str[], int l, int r) {
    if (l == r) {
        printf("%s\n", str); // Print the permutation
    } else {
        for (int i = l; i <= r; i++) {
            swap(str, l, i); // Swap the current index with the first
            permutations(str, l + 1, r); // Generate permutations for
the rest
            swap(str, l, i); // Restore the original string order
        }
    }
}

int main() {
    char str[] = "ABC";
    int n = strlen(str);
    permutations(str, 0, n - 1);
    return 0;
}
```

5. Determine Whether a Given String is a Palindrome

Pseudocode:

c

Copy code

```
// Initialize string and start and end indices
char str[] = "madam";
int start = 0, end = strlen(str) - 1;

// Check if the string is a palindrome
while (start < end) {
    if (str[start] != str[end]) {
        print("Not a palindrome");
        return;
    }
    start++;
    end--;
}
print("Palindrome");
```

Code:

c

Copy code

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

int main() {
    char str[] = "madam";
    int start = 0, end = strlen(str) - 1;

    // Check if the string is a palindrome
    while (start < end) {
        if (str[start] != str[end]) {
            printf("Not a palindrome\n");
            return 0;
        }
        start++;
        end--;
    }

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

