Programming in C

Practice Problems with Solutions — Part 3

Chapter: Arrays and Strings — For Engineering / Polytechnic Students

This is a further continuation of the Arrays and Strings practice problem set, numbered 31 onward so it can be used alongside Parts 1 and 2. It covers array copying, matrix row/column and diagonal sums, binary search, and further string-processing exercises, including manual versions of strcpy() and strcat().

Section A: Arrays

Problem 31: Copy Elements of One Array into Another

Problem Statement:

Write a C program to copy all the elements of one array into another array of the same size.

Solution:

```
#include <stdio.h>
int main()
{
    int source[50], dest[50], n, i;

    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter %d elements: ", n);
    for (i = 0; i < n; i++)
        scanf("%d", &source[i]);

    for (i = 0; i < n; i++)
        dest[i] = source[i];

    printf("Copied array: ");
    for (i = 0; i < n; i++)
        printf("%d ", dest[i]);
    printf("\n");
    return 0;
}
```

Sample Output:

```
Enter number of elements: 4
Enter 4 elements: 7 3 9 1
Copied array: 7 3 9 1
```

Explanation:

Since C does not allow copying arrays with a simple assignment such as dest = source, the loop explicitly copies each element from source to the same index in dest, one position at a time.

Problem 32: Row-wise and Column-wise Sum of a Matrix

Problem Statement:

Write a C program to calculate and print the sum of each row and the sum of each column of a given M x N matrix.

Solution:

```
#include <stdio.h>
int main()
{
    int a[10][10], rows, cols, i, j, sum;

    printf("Enter rows and columns: ");
    scanf("%d %d", &rows, &cols);

    printf("Enter matrix elements:\n");
    for (i = 0; i < rows; i++)
        for (j = 0; j < cols; j++)
            scanf("%d", &a[i][j]);

    for (i = 0; i < rows; i++)
    {
        sum = 0;
        for (j = 0; j < cols; j++)
            sum = sum + a[i][j];
        printf("Sum of row %d = %d\n", i + 1, sum);
    }

    for (j = 0; j < cols; j++)
    {
        sum = 0;
        for (i = 0; i < rows; i++)
            sum = sum + a[i][j];
        printf("Sum of column %d = %d\n", j + 1, sum);
    }
    return 0;
}
```

Sample Output:

```
Sum of row 1 = 6
Sum of row 2 = 15
Sum of column 1 = 5
Sum of column 2 = 7
Sum of column 3 = 9
```

Explanation:

The first nested loop pair fixes a row i and adds up all its columns before moving to the next row. The second nested loop pair does the reverse, fixing a column j and adding up all its rows, so together they produce both sets of totals.

Problem 33: Sum of Both Diagonals of a Square Matrix

Problem Statement:

Write a C program to find the sum of the principal diagonal and the secondary diagonal of a square (N x N) matrix.

Solution:

```
#include <stdio.h>
int main()
```

```
{
    int a[10][10], n, i, j;
    int principalSum = 0, secondarySum = 0;

    printf("Enter size of the square matrix: ");
    scanf("%d", &n);

    printf("Enter matrix elements:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &a[i][j]);

    for (i = 0; i < n; i++)
    {
        principalSum = principalSum + a[i][i];
        secondarySum = secondarySum + a[i][n - 1 - i];
    }

    printf("Principal diagonal sum = %d\n", principalSum);
    printf("Secondary diagonal sum = %d\n", secondarySum);
    return 0;
}
```

Sample Output:

```
Enter size of the square matrix: 3
Principal diagonal sum = 15
Secondary diagonal sum = 15
```

Explanation:

The principal diagonal elements satisfy row index equal to column index, that is a[i][i]. The secondary diagonal runs the other way, so its elements satisfy column index equal to n - 1 - i, that is a[i][n-1-i]. A single loop computes both sums together.

Problem 34: Count Even and Odd Numbers in an Array

Problem Statement:

Write a C program to count how many even numbers and how many odd numbers are present in a given array.

Solution:

```
#include <stdio.h>
int main()
{
    int arr[50], n, i, evenCount = 0, oddCount = 0;

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

    for (i = 0; i < n; i++)
    {
        if (arr[i] % 2 == 0)
            evenCount++;
        else
            oddCount++;
    }
```

```
    printf("Even numbers = %d\n", evenCount);
    printf("Odd numbers = %d\n", oddCount);
    return 0;
}
```

Sample Output:

```
Enter number of elements: 6
Enter 6 elements: 4 7 10 13 8 5
Even numbers = 3
Odd numbers = 3
```

Explanation:

The loop checks each array element's remainder when divided by 2, using the if-else statement to increment the correct counter: evenCount for a remainder of 0, and oddCount otherwise.

Problem 35: Sum of All Elements in a Two Dimensional Array

Problem Statement:

Write a C program to find the sum of all the elements stored in a two dimensional (M x N) array.

Solution:

```
#include <stdio.h>
int main()
{
    int a[10][10], rows, cols, i, j, sum = 0;

    printf("Enter rows and columns: ");
    scanf("%d %d", &rows, &cols);

    printf("Enter matrix elements:\n");
    for (i = 0; i < rows; i++)
    {
        for (j = 0; j < cols; j++)
        {
            scanf("%d", &a[i][j]);
            sum = sum + a[i][j];
        }
    }

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

Sample Output:

```
Enter rows and columns: 2 3
Enter matrix elements:
1 2 3 4 5 6
Sum of all elements = 21
```

Explanation:

The nested loop visits every element of the two dimensional array exactly once, row by row, and adds each one to sum as soon as it is read, giving the combined total of the entire matrix.

Problem 36: Binary Search in a Sorted Array

Problem Statement:

Write a C program to search for a given element in a sorted array using the binary search technique.

Solution:

```
#include <stdio.h>
int main()
{
    int arr[50], n, i, key, low, high, mid, found = 0;

    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter %d elements in sorted (ascending) order: ", n);
    for (i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    printf("Enter element to search: ");
    scanf("%d", &key);

    low = 0;
    high = n - 1;
    while (low <= high)
    {
        mid = (low + high) / 2;
        if (arr[mid] == key)
        {
            printf("Element found at position %d\n", mid + 1);
            found = 1;
            break;
        }
        else if (arr[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }

    if (!found)
        printf("Element not found in the array.\n");

    return 0;
}
```

Sample Output:

```
Enter number of elements: 6
Enter 6 elements in sorted (ascending) order: 2 5 8 12 16 23
Enter element to search: 12
Element found at position 4
```

Explanation:

Binary search repeatedly checks the middle element of the current search range. If the middle element matches key, the search stops; if key is larger, the search continues only in the right half (low = mid + 1); if key is smaller, it continues only in the left half (high = mid - 1). Because the array is sorted, each comparison eliminates half of the remaining elements, making this much faster than a linear search on large arrays.

Section B: Strings

Problem 37: Count the Number of Words in a Sentence

Problem Statement:

Write a C program to count the number of words in a sentence, assuming words are separated by single spaces.

Solution:

```
#include <stdio.h>
int main()
{
    char str[200];
    int i, wordCount = 0;

    printf("Enter a sentence: ");
    fgets(str, sizeof(str), stdin);

    for (i = 0; str[i] != '\0'; i++)
    {
        if (str[i] == ' ' && str[i + 1] != ' ' && str[i + 1] != '\0')
            wordCount++;
    }
    wordCount++;   // last word has no trailing space before it

    printf("Number of words = %d\n", wordCount);
    return 0;
}
```

Sample Output:

```
Enter a sentence: Programming in C is fun
Number of words = 5
```

Explanation:

fgets() is used instead of scanf("%s", ...) so that the full sentence, including spaces, is read as one line. The loop counts a new word every time a space is followed by a non-space character, and one is added at the end to account for the final word.

Problem 38: Check Whether a String Contains Only Alphabets

Problem Statement:

Write a C program to check whether a given string consists only of alphabet characters (A-Z, a-z).

Solution:

```
#include <stdio.h>
int main()
{
    char str[100];
    int i, isAlpha = 1;

    printf("Enter a string: ");
    scanf("%s", str);

    for (i = 0; str[i] != '\0'; i++)
    {
        if (!((str[i] >= 'a' && str[i] <= 'z') || (str[i] >= 'A' && str[i] <= 'Z')))
        {
            isAlpha = 0;
```

```
            break;
        }
    }

    if (isAlpha)
        printf("The string contains only alphabets.\n");
    else
        printf("The string contains non-alphabet characters.\n");

    return 0;
}
```

Sample Output:

```
Enter a string: Polytechnic
The string contains only alphabets.
```

Explanation:

Each character is tested against both the lowercase and uppercase letter ranges. As soon as a character fails both range checks, isAlpha is set to 0 and break exits the loop immediately, since even one non-letter character disqualifies the whole string.

Problem 39: Convert a Word to Proper Case (First Letter Uppercase, Rest Lowercase)

Problem Statement:

Write a C program to convert a single word so that only its first letter is uppercase and every other letter is lowercase.

Solution:

```
#include <stdio.h>
int main()
{
    char str[100];
    int i;

    printf("Enter a word: ");
    scanf("%s", str);

    if (str[0] >= 'a' && str[0] <= 'z')
        str[0] = str[0] - 32;

    for (i = 1; str[i] != '\0'; i++)
    {
        if (str[i] >= 'A' && str[i] <= 'Z')
            str[i] = str[i] + 32;
    }

    printf("Proper case: %s\n", str);
    return 0;
}
```

Sample Output:

```
Enter a word: eNGINEERING
Proper case: Engineering
```

Explanation:

The first character is separately converted to uppercase if it is currently lowercase. Every subsequent character is then converted to lowercase if it is currently uppercase, so the final result always has exactly one capital letter, at the very start of the word.

Problem 40: Count Occurrences of a Word in a Sentence

Problem Statement:

Write a C program to count how many times a specific word appears in a given sentence.

Solution:

```
#include <stdio.h>
#include <string.h>
int main()
{
    char sentence[200], word[50], token[50];
    int i = 0, j = 0, count = 0;

    printf("Enter a sentence: ");
    fgets(sentence, sizeof(sentence), stdin);
    printf("Enter word to count: ");
    scanf("%s", word);

    while (sentence[i] != '\0')
    {
        if (sentence[i] != ' ' && sentence[i] != '\n')
        {
            token[j] = sentence[i];
            j++;
        }
        else
        {
            token[j] = '\0';
            if (strcmp(token, word) == 0)
                count++;
            j = 0;
        }
        i++;
    }

    printf("'%s' occurs %d time(s) in the sentence.\n", word, count);
    return 0;
}
```

Sample Output:

```
Enter a sentence: the cat sat on the mat
Enter word to count: the
'the' occurs 2 time(s) in the sentence.
```

Explanation:

The sentence is scanned one character at a time and built up into token until a space or newline marks the end of a word; at that point, token is compared against word using strcmp(), and count is incremented on an exact match, before token is reset to build the next word.

Problem 41: Concatenate Two Strings Without Using strcat()

Problem Statement:

Write a C program to concatenate two strings without using the standard library function strcat().

Solution:

```
#include <stdio.h>
int main()
{
    char str1[100], str2[50];
    int i, j;

    printf("Enter first string: ");
    scanf("%s", str1);
    printf("Enter second string: ");
    scanf("%s", str2);

    // find the end of the first string
    for (i = 0; str1[i] != '\0'; i++)
        ;

    // copy the second string right after it
    for (j = 0; str2[j] != '\0'; j++, i++)
        str1[i] = str2[j];
    str1[i] = '\0';

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

Sample Output:

```
Enter first string: Web
Enter second string: ByArun
Concatenated string: WebByArun
```

Explanation:

The first loop does nothing except advance i to the index of str1's null terminator, effectively finding where str1 ends. The second loop then copies str2 character by character starting from that position, and a fresh null terminator is placed once both strings have been combined.

Problem 42: Copy One String into Another Without Using strcpy()

Problem Statement:

Write a C program to copy the contents of one string into another string without using the standard library function strcpy().

Solution:

```
#include <stdio.h>
int main()
{
    char source[100], dest[100];
    int i;

    printf("Enter a string: ");
    scanf("%s", source);

    for (i = 0; source[i] != '\0'; i++)
        dest[i] = source[i];
    dest[i] = '\0';
```

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

Sample Output:

```
Enter a string: polynoteshub
Copied string: polynoteshub
```

Explanation:

The loop copies each character of source into dest at the same index, stopping as soon as it reaches the null terminator of source. Once the loop ends, a null terminator is placed at the corresponding position in dest, so it becomes a properly terminated copy of the original string.