Programming in C

Practice Problems with Solutions — Part 4

Chapter: Decision Control and Looping Statements — For Engineering / Polytechnic Students

This is a further continuation of the practice problem set, numbered 43 onward so it can be used alongside Parts 1, 2, and 3. It covers additional decision-making, looping, array, and string problems, including character classification, Pascal's triangle, array frequency and merging, and further string-processing exercises.

Section A: Decision Making and Branching Statements

Problem 43: Classify a Character (Alphabet, Digit, or Special Character)

Problem Statement:

Write a C program to read a character and determine whether it is an alphabet, a digit, or a special character, using nested if-else statements.

Solution:

```
#include <stdio.h>
int main()
{
    char ch;
    printf("Enter a character: ");
    scanf("%c", &ch);

    if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))
        printf("%c is an alphabet.\n", ch);
    else
    {
        if (ch >= '0' && ch <= '9')
            printf("%c is a digit.\n", ch);
        else
            printf("%c is a special character.\n", ch);
    }
    return 0;
}
```

Sample Output:

```
Enter a character: 7
7 is a digit.
```

Explanation:

The outer if checks whether ch falls in either letter range. If not, the nested if-else inside the else block further separates digits ('0' to '9') from every other symbol, which is treated as a special character.

Problem 44: Electricity Bill Calculation Using Slab Rates

Problem Statement:

Poly Notes Hub

Write a C program to calculate an electricity bill based on the number of units consumed, using the following slabs: up to 100 units - Rs. 5 per unit, next 100 units (101-200) - Rs. 7 per unit, above 200 units - Rs. 10 per unit. Use an else-if ladder to apply the correct rate.

Solution:

```
#include <stdio.h>
int main()
{
    int units;
    float bill;

    printf("Enter units consumed: ");
    scanf("%d", &units);

    if (units <= 100)
        bill = units * 5.0;
    else if (units <= 200)
        bill = (100 * 5.0) + (units - 100) * 7.0;
    else
        bill = (100 * 5.0) + (100 * 7.0) + (units - 200) * 10.0;

    printf("Electricity bill = Rs. %.2f\n", bill);
    return 0;
}
```

Sample Output:

```
Enter units consumed: 250
Electricity bill = Rs. 2200.00
```

Explanation:

The else-if ladder checks which slab the total number of units falls into. For units beyond the first 100 or 200, the bill is built up in stages — the earlier slabs are always charged at their own fixed rate, and only the remaining units are charged at the next higher rate.

Section B: Loop Statements

Problem 45: Check Whether a Number is a Perfect Number

Problem Statement:

Write a C program to check whether a given number is a perfect number, that is, the sum of its proper divisors (excluding itself) equals the number itself.

Solution:

```
#include <stdio.h>
int main()
{
    int n, i, sum = 0;
    printf("Enter a number: ");
    scanf("%d", &n);

    for (i = 1; i < n; i++)
    {
        if (n % i == 0)
            sum = sum + i;
```

Poly Notes Hub

```
    }

    if (sum == n)
        printf("%d is a perfect number.\n", n);
    else
        printf("%d is not a perfect number.\n", n);

    return 0;
}
```

Sample Output:

```
Enter a number: 28
28 is a perfect number.
```

Explanation:

The for loop checks every integer from 1 up to n-1 to see if it divides n exactly (n % i == 0); every such divisor is added to sum. If this sum of proper divisors equals n itself, the number is perfect — as with 28, whose divisors 1, 2, 4, 7, and 14 add up to 28.

Problem 46: Print Pascal's Triangle

Problem Statement:

Write a C program to print Pascal's Triangle up to a given number of rows, using nested loops.

Solution:

```
#include <stdio.h>
int main()
{
    int rows, i, j, coef;
    printf("Enter number of rows: ");
    scanf("%d", &rows);

    for (i = 0; i < rows; i++)
    {
        coef = 1;
        for (j = 0; j <= i; j++)
        {
            printf("%d ", coef);
            coef = coef * (i - j) / (j + 1);
        }
        printf("\n");
    }
    return 0;
}
```

Sample Output:

```
Enter number of rows: 5
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Explanation:

Poly Notes Hub

For each row i, coef starts at 1 (the first binomial coefficient of that row) and is updated after every print using the standard relation coef = coef * (i - j) / (j + 1), which generates the next coefficient in the row without needing a separate factorial calculation.

Problem 47: Count the Number of Digits in a Number

Problem Statement:

Write a C program to count the number of digits present in a given integer using a while loop.

Solution:

```
#include <stdio.h>
int main()
{
    int num, count = 0;
    printf("Enter a number: ");
    scanf("%d", &num);

    if (num == 0)
        count = 1;

    while (num != 0)
    {
        num = num / 10;
        count++;
    }

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

Sample Output:

```
Enter a number: 30427
Number of digits = 5
```

Explanation:

On each pass, the number is divided by 10 (discarding the last digit) and count is incremented, until num finally reaches zero. The special case for 0 is handled separately, since the loop would otherwise never execute and report a count of zero digits.

Problem 48: Sum of the Series 1 Squared + 2 Squared + ... + N Squared

Problem Statement:

Write a C program to calculate the sum of the series 1 squared + 2 squared + 3 squared + ... + N squared using a for loop.

Solution:

```
#include <stdio.h>
int main()
{
    int n, i;
    long sum = 0;

    printf("Enter N: ");
    scanf("%d", &n);
```

Poly Notes Hub

```
    for (i = 1; i <= n; i++)
        sum = sum + (i * i);

    printf("Sum of series = %ld\n", sum);
    return 0;
}
```

Sample Output:

```
Enter N: 5
Sum of series = 55
```

Explanation:

The for loop runs i from 1 to n, and on each pass adds the square of the current value of i to the running total sum, building up the complete series total by the time the loop ends.

Section C: Arrays

Problem 49: Frequency of Each Element in an Array

Problem Statement:

Write a C program to find and display the frequency (number of occurrences) of each distinct element in an array.

Solution:

```
#include <stdio.h>
int main()
{
    int arr[50], visited[50] = {0}, n, i, j, count;

    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 (visited[i] == 1)
            continue;   // this value's frequency was already printed

        count = 1;
        for (j = i + 1; j < n; j++)
        {
            if (arr[i] == arr[j])
            {
                visited[j] = 1;
                count++;
            }
        }
        printf("%d occurs %d time(s)\n", arr[i], count);
    }
    return 0;
}
```

Sample Output:

Poly Notes Hub

```
Enter number of elements: 6
Enter 6 elements: 4 2 4 5 2 4
4 occurs 3 time(s)
2 occurs 2 time(s)
5 occurs 1 time(s)
```

Explanation:

The visited array marks positions whose value has already been counted and printed, so continue is used to skip them. For every new (unvisited) value, the inner loop scans the rest of the array, marking and counting every further occurrence of the same value.

Problem 50: Merge Two Arrays into a Third Array

Problem Statement:

Write a C program to merge two arrays into a single third array by placing all elements of the first array followed by all elements of the second array.

Solution:

```c
#include <stdio.h>
int main()
{
    int a[30], b[30], merged[60];
    int n1, n2, i;

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

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

    for (i = 0; i < n1; i++)
        merged[i] = a[i];
    for (i = 0; i < n2; i++)
        merged[n1 + i] = b[i];

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

Sample Output:

```
Merged array: 1 3 5 2 4 6
```

Explanation:

The first loop copies all n1 elements of a directly into the start of merged. The second loop copies the n2 elements of b into merged as well, but offsets each index by n1 so that they are placed immediately after the elements already copied from a.

Problem 51: 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.

Section D: Strings

Problem 52: Convert an Uppercase String to Lowercase

Problem Statement:

Poly Notes Hub

Write a C program to convert an uppercase string to lowercase without using the standard library function tolower().

Solution:

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

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

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

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

Sample Output:

```
Enter a string: POLYNOTESHUB
Lowercase string: polynoteshub
```

Explanation:

Since every uppercase letter is exactly 32 positions before its lowercase equivalent in the ASCII table, adding 32 to any character in the range 'A' to 'Z' converts it directly to its lowercase form, modifying the character array in place.

Problem 53: Check if One String is a Substring of Another

Problem Statement:

Write a C program to check whether a given string appears as a substring inside another string, without using the library function strstr().

Solution:

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str[100], sub[50];
    int i, j, lenStr, lenSub, found = 0;

    printf("Enter main string: ");
    scanf("%s", str);
    printf("Enter substring to search: ");
    scanf("%s", sub);

    lenStr = strlen(str);
    lenSub = strlen(sub);

    for (i = 0; i <= lenStr - lenSub; i++)
    {
```

Poly Notes Hub

```
            for (j = 0; j < lenSub; j++)
            {
                if (str[i + j] != sub[j])
                    break;
            }
            if (j == lenSub)
            {
                found = 1;
                break;
            }
        }

        if (found)
            printf("Substring found at position %d\n", i + 1);
        else
            printf("Substring not found.\n");

        return 0;
    }
```

Sample Output:

```
Enter main string: programming
Enter substring to search: gram
Substring found at position 4
```

Explanation:

The outer loop tries every possible starting position i in str where sub could fit, and the inner loop compares sub character by character against str starting at that position. If the inner loop runs all the way to lenSub without a mismatch, the substring has been found.

Problem 54: 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, k, 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
```

Poly Notes Hub

```
            {
                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.

Poly Notes Hub