5.1 A barcode scanner for Universal Product Codes (UPCs) verifies the 12-digit code scanned by comparing the code's last digit (called a check digit) to its own computation of the check digit from the first 11 digits as follows:

1. Calculate the sum of the digits in the odd-numbered positions (the first, third, ..., eleventh digits) and multiply this sum by 3.
2. Calculate the sum of the digits in the even-numbered positions (the second, fourth, ..., tenth digits) and add this to the previous result.
3. If the last digit of the result from step 2 is 0, then 0 is the check digit. Otherwise, subtract the last digit from 10 to calculate the check digit.
4. If the check digit matches the final digit of the 12-digit UPC, the UPC is assumed correct.

Write a program that prompts the user to enter the 12 digits of a barcode separated by spaces. The program should store the digits in an integer array, calculate the check digit, and compare it to the final barcode digit. If the digits match, output the barcode with the message "validated." If not, output the barcode with the message "error in barcode." Also, output with labels the results from steps 1 and 2 of the check-digit calculations. Note that the "first" digit of the barcode will be stored in element 0 of the array. Try your program on the following barcodes, three of which are valid. For the first barcode, the result from step 2 is 79 (0 + 9 + 0 + 8 + 4 + 0) * 3 + (7 + 4 + 0 + 0 + 5).

079400804501
024000162860
011110856807
051000138101

PROBLEM: Barcode Scanner for Universal Product Codes (UPCs)

IMPLEMENTATION:

```
#include<stdio.h>
 int main() {
    int barcode[12], i, step1_sum = 0, step2_sum = 0, computed_check_digit;
    printf("Enter 12 digits of the barcode separated by spaces: ");     for (i = 0; i < 12; i++) {
scanf("%d", &barcode[i]);     }
    for (i = 0; i < 11; i++) {
        if (i % 2 == 0) step1_sum += barcode[i]; // Odd positions         else step2_sum +=
barcode[i]; // Even positions     }
    step1_sum *= 3; // Multiply odd sum by 3     step2_sum+= step1_sum; //
Add even sum
    computed_check_digit = (step2_sum % 10 == 0) ? 0 : 10 - (step2_sum % 10);
    printf("Step 1 result: %d\n", step1_sum);
printf("Step 2 result: %d\n", step2_sum);
    if (computed_check_digit == barcode[11]) {
        printf("Barcode %d%d%d%d%d%d%d%d%d%d%d%d validated.\n", barcode[0], barcode[1],
barcode[2], barcode[3], barcode[4], barcode[5], barcode[6], barcode[7], barcode[8], barcode[9],
barcode[10], barcode[11]);
    } else {
        printf("Barcode %d%d%d%d%d%d%d%d%d%d%d%d error in barcode.\n", barcode[0],
barcode[1], barcode[2], barcode[3], barcode[4], barcode[5], barcode[6], barcode[7], barcode[8],
barcode[9], barcode[10], barcode[11]);     }
}
```

Input :
Enter 12 digits of the barcode separated by spaces: 0 7 9 4 0 0 8 0 4 5 0 1

Output :
Step 1 result: 7 9
Step 2 result: 1 04
Barcode 079400804501 validated.

5.2 Write a program to take two numerical lists of the same length ended by a sentinel value and store the lists in arrays x and y, each of which has 20 elements. Let n be the actual number of data values in each list. Store the product of corresponding elements of x and y in a third array, z, also of size 20. Display the arrays x, y, and z in a threecolumn table. Then compute and display the square root of the sum of the items in z.

PROBLEM: Numerical List Product and Square Root of Sum
IMPLEMENTATION:
```
#include<stdio.h>
#include<math.h>
 int main() {
    int x[20], y[20], z[20], n = 0, i;     double sum = 0.0;
    printf("Enter values for list x, terminated by -1: ");     for (i = 0; i < 20; i++) {
scanf("%d", &x[i]);         if (x[i] == -1) break;
    }
n = i;
    printf("Enter values for list y, terminated by -1: ");     for (i = 0; i < n; i++) {
scanf("%d", &y[i]);
    }
    for (i = 0; i < n; i++) {         z[i] = x[i] *
y[i];
sum += z[i];     }
    printf("x\t y\t z\n");     for (i = 0; i < n;
i++) {
        printf("%d\t %d\t %d\n", x[i], y[i], z[i]);     }
    printf("Square root of sum of z: %.2f\n", sqrt(sum));
}
```
Input:
Enter elements of the first array (terminated by -1): 2 3 4 5 -1
Enter elements of the second array (terminated by -1): 6 7 8 9 -1
Output:
Array X: 2 3 4 5
Array Y: 6 7 8 9
Array Z: 12 21 32 45
Square root of the sum of Z: 8.54

5.3 Write and test a function deblank that takes a string output and a string input argument and returns a copy of the input argument with all blanks removed.

PROBLEM: Remove Blanks from String
IMPLEMENTATION:
```
#include<stdio.h>
```

```
#include<string.h>
void deblank(char output[], const char input[]) {     int i, j = 0;
    for (i = 0; input[i] != '\0'; i++) {         if (input[i] != ' ') {
output[j++] = input[i];
        }     }
    output[j] = '\0';
}  int main() {
    char input[100], output[100];
    printf("Enter a string: ");
fgets(input, sizeof(input), stdin);     deblank(output, input);
    printf("String without blanks: %s\n", output);

    return 0;
}
```

Input:
Enter a string with spaces: Hello World! Programming in C.

Output:
String without spaces: HelloWorld!ProgramminginC.

5.4 Write and test a recursive function that returns the value of the following recursive definition:
f(x) = 0 if x ≤ 0
f(x) = f(x – 1) + 2 otherwise
What set of numbers is generated by this definition?

PROBLEM: Recursive Function Definition
IMPLEMENTATION:

```
#include<stdio.h>
 int f(int x) {
if (x <= 0) return 0;     return f(x - 1)
+ 2;
}  int main()
{     int x;
    printf("Enter a value for x: ");
scanf("%d", &x);

    printf("f(%d) = %d\n", x, f(x));
     return 0;
}
```

Input:
Enter a value for x: 5

Output:
Recursive result for f(5): 10
Generated set of numbers: 0, 2, 4, 6, 8, 10