1) Constant Variable Declaration

Objective: Learn to declare and initialize constant variables.
Write a program that declares a constant integer variable for the value of Pi (3.14) and prints it. Ensure that any attempt to modify this variable results in a compile-time error.

ANSWER:

```
#include <stdio.h>
int main() {
    const float PI = 3.14;
    printf("The value of Pi is: %f\n", PI);
    // PI = 3.14159;
    return 0;
}
```

2) Using const with Pointers

Objective: Understand how to use const with pointers to prevent modification of pointed values.
Create a program that uses a pointer to a constant integer. Attempt to modify the value through the pointer and observe the compiler’s response.

ANSWER:

```
#include <stdio.h>
int main() {
    const int value = 10;
    const int *ptr = &value;
    printf("The value pointed to by ptr is: %d\n", *ptr);
    // Attempt to modify the value through the pointer (this will cause a compile-time error)
    // *ptr = 20;
    return 0;
}
```

3)Constant Pointer

Objective: Learn about constant pointers and their usage.
Write a program that declares a constant pointer to an integer and demonstrates that you cannot change the address stored in the pointer.

ANSWER:

```
#include <stdio.h>

int main() {
    int value1 = 10;
    int value2 = 20;
    int *const ptr = &value1;
    printf("The value pointed to by ptr is: %d\n", *ptr);
    *ptr = 15;
    printf("The modified value pointed to by ptr is: %d\n", *ptr);
    // Attempt to change the address stored in the constant pointer (this will cause a compile-time error)
    // ptr = &value2;  // Uncommenting this line will result in a compile-time error
    return 0;
}
```

4) Constant Pointer to Constant Value

Objective: Combine both constant pointers and constant values.
Create a program that declares a constant pointer to a constant integer. Demonstrate that neither the pointer nor the value it points to can be changed.

ANSWER:

```
#include <stdio.h>

int main() {
    const int value = 10;
    const int *const ptr = &value;
    printf("The value pointed to by ptr is: %d\n", *ptr);

    // Attempt to modify the value through the pointer (this will cause a compile-time error)
    // *ptr = 20;  // Uncommenting this line will result in a compile-time error

    // Attempt to change the address stored in the pointer (this will also cause a compile-time error)
    // int anotherValue = 30;
    // ptr = &anotherValue;  // Uncommenting this line will result in a compile-time error

    return 0;
}
```

5) Using const in Function Parameters
Objective: Understand how to use const with function parameters.
Write a function that takes a constant integer as an argument and prints its value. Attempting to modify this parameter inside the function should result in an error.

ANSWER:

```
#include <stdio.h>
void printValue(const int num) {
    printf("The value is: %d\n", num);

    // Attempt to modify the parameter (this will cause a compile-time error)
    // num = 20;  // Uncommenting this line will result in a compile-time error
}

int main() {
    int value = 10;
    printValue(value);

    return 0;
}
```

6) Array of Constants
Objective: Learn how to declare and use arrays with const.
Create an array of constants representing days of the week. Print each day using a loop, ensuring that no modifications can be made to the array elements.

ANSWER:

```
#include <stdio.h>

int main() {
    const char *daysOfWeek[] = {
        "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"
    };

       int numDays = sizeof(daysOfWeek) / sizeof(daysOfWeek[0]);
       for (int i = 0; i < numDays; i++) {
        printf("%s\n", daysOfWeek[i]);
    }

    // Attempt to modify an element in the array (this will cause a compile-time error)
    // daysOfWeek[0] = "Funday";  // Uncommenting this line will result in a compile-time error

    return 0;
}
```

7)Constant Expressions
Objective: Understand how constants can be used in expressions.
Write a program that uses constants in calculations, such as calculating the area of a circle using const.

ANSWER:

```
#include <stdio.h>
int main() {
    const float PI = 3.14;
    const float radius = 5.0;
    float area = PI * radius * radius;
    printf("The area of the circle with radius %.2f is: %.2f\n", radius, area);
    return 0;
}
```

8)Constant Variables in Loops
Objective: Learn how constants can be used within loops for fixed iterations.
Create a program that uses a constant variable to define the number of iterations in a loop, ensuring it cannot be modified during execution.

ANSWER:

```
#include <stdio.h>
int main() {
    const int NUM_ITERATIONS = 5;
    for (int i = 0; i < NUM_ITERATIONS; i++) {
        printf("Iteration %d\n", i + 1);
    }
    // Attempting to modify NUM_ITERATIONS will cause a compile-time error
    // NUM_ITERATIONS = 10;  // Uncommenting this line will result in a compile-time error
    return 0;
}
```

9)Constant Global Variables
Objective: Explore global constants and their accessibility across functions.
Write a program that declares a global constant variable and accesses it from multiple functions without modifying its value.

ANSWER:

```
#include <stdio.h>
const int MAX_VALUE = 100;
void printMaxValue() {
    printf("The maximum value is: %d\n", MAX_VALUE);
}
```

```
// Function to check if a given number is less than the maximum value
void checkValue(int value) {
    if (value < MAX_VALUE) {
        printf("%d is less than the maximum value.\n", value);
    } else {
        printf("%d is greater than or equal to the maximum value.\n", value);
    }
}

int main() {
    // Access the global constant variable in the main function
    printf("Global constant MAX_VALUE in main: %d\n", MAX_VALUE);

    // Call other functions that access the global constant
    printMaxValue();
    checkValue(50);
    checkValue(150);

    // Attempting to modify the global constant will cause a compile-time error
    // MAX_VALUE = 200;  // Uncommenting this line will result in a compile-time error

    return 0;
}
```

11)Initializing Arrays

Requirements

In this challenge, you are going to create a program that will find all the prime numbers from 3-100 there will be no input to the program

•The output will be each prime number separated by a space on a single line

• You will need to create an array that will store each prime number as it is generated

-You can hard-code the first two prime numbers (2 and 3) in the primes array

You should utilize loops to only find prime numbers up to 100 and a loop to print out the primes array

ANSWER:

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

int main() {
    int primes[100]; // Array to store prime numbers, assuming max 100 primes for simplicity
    int count = 2;   // Initialize count to 2 as we already know the first two primes
    primes[0] = 2;   // Hard-code the first prime number
    primes[1] = 3;   // Hard-code the second prime number

    for (int i = 5; i <= 100; i += 2) { // Skip even numbers, start from 5
        bool isPrime = true;
        for (int j = 1; primes[j] * primes[j] <= i; j++) {
            if (i % primes[j] == 0) {
                isPrime = false;
                break;
            }
        }

        if (isPrime) {
            primes[count] = i;
            count++;
        }
    }
    for (int i = 0; i < count; i++) {
        printf("%d ", primes[i]);
    }
    printf("\n");

    return 0;
}
```

12) Create a program that reverses the elements of an array. Prompt the user to enter values and print both the original and reversed arrays.

ANSWER:

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        printf("Element %d: ", i + 1);
        scanf("%d", &arr[i]);
    }
    printf("\nOriginal array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    for (int i = 0; i < n / 2; i++) {
        int temp = arr[i];
        arr[i] = arr[n - i - 1];
        arr[n - i - 1] = temp;
    }
    printf("\nReversed array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }

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

13)Program 1: Find the Maximum Element in an Array

ANSWER:

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        printf("Element %d: ", i + 1);
        scanf("%d", &arr[i]);
    }

    int max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }

    printf("The maximum element is: %d\n", max);

    return 0;
}
```

14)Program 2: Count Occurrences of a Specific Integer in an Array

ANSWER:

```
#include <stdio.h>

int main() {
    int n, search, count = 0;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        printf("Element %d: ", i + 1);
        scanf("%d", &arr[i]);
    }
    printf("Enter the integer to count occurrences of: ");
    scanf("%d", &search);
    for (int i = 0; i < n; i++) {
        if (arr[i] == search) {
            count++;
        }
    }
    printf("The integer %d appears %d times in the array.\n", search, count);

    return 0;
}
```

15)Requirements

In this challenge, you are to create a C program that uses a two-dimensional array in a weather program.

- This program will find the total rainfall for each year, the average yearly rainfall, and the average rainfall for each month

- Input will be a 2D array with hard-coded values for rainfall amounts for the past 5 years

- The array should have 5 rows and 12 columns. Rainfall amounts can be floating point numbers

Example output

```
YEAR   RAINFALL (inches)

2010  32.4

2011  37.9

2012  49.8

2013  44.0

2014  32.9

The yearly average is 39.4 inches.

MONTHLY AVERAGES:

Jan-7.3
Feb-7.3
Mar-4.9
Apr-3.0
May-2.3
Jun-0.6
Jul-1.2
Aug-0.3
Sep-0.5
Oct-1.7
Nov-3.6
Dec-6.7
```

ANSWER:

```c
#include <stdio.h>

#define YEARS 5
#define MONTHS 12

int main() {
    float rainfall[YEARS][MONTHS];
    float yearlyTotals[YEARS] = {0};
    float totalRainfall = 0;
    printf("Enter the rainfall data for each month (in inches):\n");
    for (int year = 0; year < YEARS; year++) {
        printf("Year 201%d:\n", year);
        for (int month = 0; month < MONTHS; month++) {
            printf("  Month %d: ", month + 1);
            scanf("%f", &rainfall[year][month]);
            yearlyTotals[year] += rainfall[year][month];
        }
        totalRainfall += yearlyTotals[year];
    }
    printf("\nYEAR    RAINFALL (inches)\n");
    for (int year = 0; year < YEARS; year++) {
        printf("201%d    %.1f\n", year, yearlyTotals[year]);
    }
    float yearlyAverage = totalRainfall / YEARS;
    printf("\nThe yearly average is %.1f inches.\n", yearlyAverage);
    printf("\nMONTHLY AVERAGES:\n");
    const char *months[MONTHS] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun",
                                  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"};

    for (int month = 0; month < MONTHS; month++) {
        float monthlyTotal = 0;
        for (int year = 0; year < YEARS; year++) {
            monthlyTotal += rainfall[year][month];
        }
        printf("%s    %.1f\n", months[month], monthlyTotal / YEARS);
    }

    return 0;
}
```