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.

```
#include <stdio.h>

int main()
{
float const pi = 3.14;
printf("The value of pi is %.2f\n", pi);
}
```

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 modity the value through the pointer and observe the compiler's response.

```
#include <stdio.h>

int main()
{
int const num = 10;

int const *ptr = &num;

//num= 5; Compiler error

//"ptr = 5; Compiler error

printf("The value of num is %d\n", num);
}
```

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

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

```
int main()
{
int a = 10;

int b =20;

int  *const ptr =  &num;

//num = 5;  //No error

//*ptr = 5;  //No error

//*ptr = &b; //Compile error

printf("The value of a is %d\n", num);
}
```

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.

```
#include <stdio.h>

int main()
{
const int num = 10;

int num1 =20;

const int *const ptr = &num;

// num = 6; //Compiler error

//*ptr=6;//Compiler error

// ptr = &num1; //Compiler error

printf("The value of num is %d\n", num);
}
```

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.

```
#include <stdio.h>

int add(const int num1, const int num2).

{

//num1 = 14;   //Compiler error
//num2 = 7;     //Compiler error

return (num1 + num2);
}

int main()
{
const int a = 10, b = 20;
int result = add(10, 20);
printf("%d", result);
}
```

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.

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

const char *const days[] = {

"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

//days[0]="apple";

for (int i=0; i<7; i++) {

printf("%s\n", days[i]);
}
```

```
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.

```
#include <stdio.h>

const float Pi = 3.14;

int main()
{
int rad = 20;

float area = Pi * rad * rad;

printf("Area of the circle is: %f\n",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.

```
#include <stdio.h>

const int counter = 10;

int main()
{
for (int i = 1; i<= counter; i++)
{
printf("counter %d\n", i);
}
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..

```
include <stdio.h>

const int num = 50;

void f1()
{

printf("The value of num is: %d\n", num);

}

void f2()
{
printf("Double the value of num is: %d\n", num * 2);
 }

int main()
 {

f1();

f2();

return 0;

}
```

10) Initializing Arrays

Requirements In this challenge, you are going to create a program that will find 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

```
#include <stdio.h>

int main() {
    int primes[50] = {2, 3};
    int count = 2;
    int isPrime;

    for (int num = 5; num <= 100; num += 2) {

        isPrime = 1;

        for (int i = 1; i < count; i++) {
            if (num % primes[i] == 0) {
                isPrime = 0;
                break;
            }
        }

                if (isPrime) {
            primes[count] = num;
            count++;
        }
    }

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

    return 0;
}
```

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

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the no 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:\n");
    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 - 1 - i];
        arr[n - 1 - i] = temp;
    }

    printf("\nReversed array:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

12) Write a program that to find the maximum element in an array of integers. The program should prompt the user for input and display the maximum value.

```
#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 in this array is: %d\n", max);

    return 0;
}
```

13) Write a program that counts and displays how many times a specific integer appears in an array entered by the user.

```
#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: ");
    scanf("%d", &search);

    for (int i = 0; i < n; i++) {
        if (arr[i] == search) {
            count++;
        }
    }

    printf(" %d appears %d time in the array.\n", search, count);
```

```
    return 0;
}
```

14) Assignment
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

```
#include <stdio.h>

int main() {

    float rainfall[5][12] = {
        {7.3, 7.3, 4.9, 3.0, 2.3, 0.6, 1.2, 0.3, 0.5, 1.7, 3.6, 6.7}, // 2010
        {8.0, 6.5, 4.5, 2.8, 2.0, 0.5, 1.1, 0.4, 0.6, 1.5, 3.5, 6.5}, // 2011
        {9.2, 8.1, 5.1, 3.5, 2.6, 1.0, 1.4, 0.7, 0.9, 1.8, 4.0, 7.5}, // 2012
        {8.5, 7.0, 5.0, 3.2, 2.4, 0.9, 1.3, 0.6, 0.8, 1.6, 3.8, 6.9}, // 2013
        {7.1, 6.8, 4.6, 3.1, 2.2, 0.7, 1.0, 0.5, 0.7, 1.4, 3.3, 6.2}  // 2014
    };

    float yearlyTotal[5] = {0};
    float monthlyAverage[12] = {0};
    float totalRainfall = 0;


    for (int year = 0; year < 5; year++) {
        for (int month = 0; month < 12; month++) {
            yearlyTotal[year] += rainfall[year][month];
            monthlyAverage[month] += rainfall[year][month];
        }
        totalRainfall += yearlyTotal[year];
    }
```

```
    for (int month = 0; month < 12; month++) {
        monthlyAverage[month] /= 5;
    }


    printf("YEAR   RAINFALL (inches)\n");
    for (int year = 0; year < 5; year++) {
        printf("201%d   %.1f\n", year, yearlyTotal[year]);
    }

    float yearlyAverage = totalRainfall / 5;
    printf("\nThe yearly average is %.1f inches.\n", yearlyAverage);

    printf("\nMONTHLY AVERAGES:\n");
    printf("Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec\n");
    for (int month = 0; month < 12; month++) {
        printf("%.1f   ", monthlyAverage[month]);
    }
    printf("\n");

    return 0;
}
```