Assignment 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() {
   const float PI = 3.14;
   printf("Value of PI: %.2f\n", PI);
   return 0;
}
```

OUTPUT
```
Value of PI: 3.14
```

Assignment 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.

```
#include <stdio.h>

int main() {
   int value = 5;
   const int *ptr = &value;
   printf("Value through ptr: %d\n", *ptr);
   return 0;
}
```

OUTPUT

```
Value through ptr: 5
```

Assignment 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 num1 = 10;
   int num2 = 20;
   int *const ptr = &num1;
```

```
    printf("Value pointed by ptr: %d\n", *ptr);
    *ptr = 30;
    printf("New value of num1: %d\n", num1);
    return 0;
}
```

OUTPUT

```
Value pointed by ptr: 10
New value of num1: 30
```

Assignment 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() {
    int num = 10;
    const int *const ptr = &num;

    printf("Value pointed by ptr: %d\n", *ptr);
    return 0;
}
```

OUTPUT

```
Value pointed by ptr: 10
```

Assignment 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>

void printConstValue(const int value) {
    printf("Constant value: %d\n", value);
}

int main() {
    int num = 10;
    printConstValue(num);
    return 0;
}
```

OUTPUT

```
Constant value: 10
```

Assignment 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 *daysOfWeek[] = {"Sunday", "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday"};

    for (int i = 0; i < 7; i++) {
        printf("%s\n", daysOfWeek[i]);
    }


    return 0;
}
```

OUTPUT

```
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
```

Assignment 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>

int main() {
    const float PI = 3.14;
    int radius = 5;
    float area = PI * radius * radius;

    printf("Area of circle: %.2f\n", area);

    return 0;
}
```

OUTPUT

```
Area of circle: 78.50
```

Assignment 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>

int main() {
    const int iterations = 5;

    for (int i = 0; i < iterations; i++) {
        printf("Iteration %d\n", i + 1);
    }

    // Uncommenting the following line will cause a compile-time error
    // iterations = 10;

    return 0;
}
```

OUTPUT

```
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
```

Assignment 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>
```

```c
const int GLOBAL_CONST = 100;

void printGlobalConst() {
  printf("Global constant in function: %d\n", GLOBAL_CONST);
}

int main() {
  printf("Global constant in main: %d\n", GLOBAL_CONST);
  printGlobalConst();


  return 0;
}
```

OUTPUT

```
Global constant in main: 100
Global constant in function: 100
```

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

```c
#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);
  for (int i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }

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

  printf("Reversed array: ");
  for (int i = n - 1; i >= 0; i--) {
    printf("%d ", arr[i]);
  }
```

```
    printf("\n");

    return 0;
}
```

OUTPUT

```
Enter the number of elements in the array: 4
Enter 4 elements: 2
3
4
6
Original array: 2 3 4 6
Reversed array: 6 4 3 2
```

11. 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);
    for (int i = 0; i < n; i++) {
        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;
}
```

OUTPUT

```
Enter the number of elements in the array: 5
Enter 5 elements: 6
```

```
2
10
8
4
The maximum element is: 10
```

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

```c
#include <stdio.h>

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

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

  printf("Enter the number to count: ");
  scanf("%d", &target);

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

  printf("The number %d appears %d times in the array.\n", target, count);

  return 0;
}
```

OUTPUT

```
Enter the number of elements in the array: 5
Enter 5 elements: 2
6
6
8
7
Enter the number to count: 6
The number 6 appears 2 times in the array.
```

13. 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

```c
#include <stdio.h>

int main() {
   int primes[100];
   int count = 2;
   primes[0] = 2;
   primes[1] = 3;

   for (int num = 5; num <= 100; num += 2) {
      int isPrime = 1;
      for (int i = 1; primes[i] * primes[i] <= num; 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]);
   }
   printf("\n");

   return 0;
}
```

OUTPUT

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

14. 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 .

```c
#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
    {7.8, 6.9, 5.0, 3.2, 2.8, 0.7, 1.3, 0.4, 0.6, 1.8, 4.0, 7.4},  // 2011
    {8.1, 7.1, 5.3, 3.5, 2.9, 0.9, 1.5, 0.7, 0.8, 1.9, 4.3, 8.2},  // 2012
    {7.6, 7.0, 5.1, 3.4, 2.7, 0.8, 1.4, 0.6, 0.7, 1.8, 4.2, 7.9},  // 2013
    {6.9, 6.8, 4.7, 3.1, 2.5, 0.5, 1.1, 0.3, 0.4, 1.5, 3.5, 6.6}   // 2014
  };

  int years = 5;
  int months = 12;
  float yearly_totals[5] = {0};
  float monthly_totals[12] = {0};
  float total_rainfall = 0;

  for (int i = 0; i < years; i++) {
    for (int j = 0; j < months; j++) {
      yearly_totals[i] += rainfall[i][j];
      monthly_totals[j] += rainfall[i][j];
    }
    total_rainfall += yearly_totals[i];
  }

  printf("Total Rainfall for Each Year:\n");
  for (int i = 0; i < years; i++) {
    printf("Year %d: %.1f inches\n", 2010 + i, yearly_totals[i]);
  }

  printf("\nAverage Yearly Rainfall: %.1f inches\n", total_rainfall / years);

  printf("\nMonthly Averages:\n");
  const char *months_names[] = {
    "Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
  };
  for (int j = 0; j < months; j++) {
    printf("%s: %.1f inches\n", months_names[j], monthly_totals[j] / years);
  }

  return 0;
}
```

OUTPUT

```
Total Rainfall for Each Year:
Year 2010: 39.4 inches
Year 2011: 41.9 inches
Year 2012: 45.2 inches
Year 2013: 43.2 inches
Year 2014: 37.9 inches

Average Yearly Rainfall: 41.5 inches

Monthly Averages:
Jan: 7.5 inches
Feb: 7.0 inches
Mar: 5.0 inches
Apr: 3.2 inches
May: 2.6 inches
Jun: 0.7 inches
Jul: 1.3 inches
Aug: 0.5 inches
Sep: 0.6 inches
Oct: 1.7 inches
Nov: 3.9 inches
Dec: 7.4 inches
```

15. Example using designated initializer.

```
#include <stdio.h>
#define MONTHS 12
int main()
{
 int days[MONTHS]={31,28,[4]=31,30,31,[1]=29};
 int i;

 for(i=0; i<MONTHS; i++)
 {
 printf("Month %d has %2d days.\n",i+1, days[i]);
 }
}
```

OUTPUT

```
Month 1 has 31 days.
Month 2 has 29 days.
Month 3 has  0 days.
Month 4 has  0 days.
Month 5 has 31 days.
Month 6 has 30 days.
Month 7 has 31 days.
Month 8 has  0 days.
Month 9 has  0 days.
Month 10 has  0 days.
Month 11 has  0 days.
```

Month 12 has 0 days.