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>

const float PI = 3.14;

int main()

{

    printf("value of Pi is %.2f",PI);

    //PI = 556.00;   //Show compiler error

    //float ptr = &PI;

    //*ptr = 556.00; //Show compiler error

    //printf("value of Pi is %.2f",PI);



}
```

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 a = 5;

    int const *ptr = &a;

    printf("001 value of a is %d\n",a);

    printf("002 value of *ptr is %d\n",*ptr);

   // *ptr = 80;//Read only data cannot be modified

    //printf("003 value of *ptr is %d\n",*ptr);

}
```

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 a = 5,b=10;

   int *const ptr = &a;

   printf("001 value of *ptr is %d\n",*ptr);

   ptr = &b;//Cannot be modified address in read only pointer

   printf("002 value of *ptr is %d\n",*ptr);

}
```

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 a = 5;

   int b = 10;

   int const *const ptr = &a;

   printf("001 value of *ptr is %d\n",*ptr);

   ptr = &b;//Cannot be modified address in read only pointer

   printf("002 value of *ptr is %d\n",*ptr);

   *ptr = 20;//Cannot modify read only data

   printf("003 value of *ptr is %d\n",*ptr);

}
```

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 printValue(const int num)

{

   printf("Value: %d\n", num);

   num = num + 10;  // Error: modifying read only data num

}

int main()

{

   const int value = 50;

   printValue(value);

   return 0;

}
```

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

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

    {

      printf("%s\n", daysOfWeek[i]);

   }
```

```
    // Attempting to modify an element in the array would cause a compilation error

    //daysOfWeek[0] = "Funday";

    return 0;

}
```

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()

{

    float const radius = 10, PI = 3.14;

    float const area = radius * radius * PI;

    printf("Area of circe is %.2f",area);

    return 0;

}
```

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 iteration = 5;

    for (int i = 0; i < iteration; i++)

    {

        printf("Iteration %d\n", i + 1);

    }
```

```
    // Attempting to modify iteration would cause a compilation error

    //NUM_ITERATIONS = 10;

    return 0;

}
```

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>
const int var = 100;

void display_var()
{
    printf("Variable value is: %d\n", var);
}


void check_positive()
{
    if(var >= 0)
    {
        printf("Variable is positive integer\n");
    }
    else
    {
        printf("Variable is negative integer\n");
    }
}

int main()
{
     printf("Variable value is: %d\n", var);


     display_var();
     check_positive();

    return 0;
}
```

Example 1 (Using array find the average of 10 user inputs)

```
#include <stdio.h>

int main()
{
    int A[10];
    int count = 10;
    long sum = 0;
```

```
    float average = 0.0f;
    printf("Enter the 10 grade\n");

    for(int i=0;i<count;i++)
    {
        printf("%2u>",i+1);
        scanf("%d",&A[i]);
        sum += A[i];
    }
    average = (float)sum/count;
    printf("\n Average of ten grades :%.2f\n",average);

    return 0;
}
```

Example 2

```
#include <stdio.h>
#define MONTHS 12
int main()
{
    //int days[MONTHS] ={31,28,31,30,31,30,31,31,30,31,30,31};
    int days[MONTHS] ={31,28,[4]=31,30,31,[1]=29};
    int i;
    for(int i=0;i<10;i++)
    {
        printf("Month %d has %2d days.\n",i+1,days[i]);
    }


    return 0;
}
```

1.WAP to find the prime numbers between 3 and 100

```
#include <stdio.h>

int main()
{
    int prime[50]={2,3};
    int i,j;
    int len=2;
    for(i=4;i<=100;i++)
    {
        int flag =1;
        for(j=2;j*j<=i;j++)
        {
            if(i%j == 0)
            {
                flag = 0;
                break;
            }

        }
        if(flag == 1)
```

```
        {
            prime[len]=i;
            len++;
        }

    }
    printf("The prime numbers are:");
    for(int k=0;k<len;k++)
    {
        printf("%d ",prime[k]);
    }

    return 0;
}
```

2.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 size;
    scanf("%d",&size);
    int array[size];
    printf("Enter %d array elements:",size);
    for(int i=0;i<size;i++)
    {
        scanf("%d",&array[i]);
    }
    printf("\nArray elements before reversing:");
    for(int i=0;i<size;i++)
    {
        printf("%d ",array[i]);
    }
    for(int i=0;i<size/2;i++)
    {
        int temp = array[i];
        array[i] = array[(size-1)-i];
        array[(size-1)-i] = temp;
    }
    printf("\nArray elements after reversing: ");
    for(int i=0;i<size;i++)
    {
        printf("%d ",array[i]);
    }


    return 0;
}
```

3.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 size;
   scanf("%d",&size);
   int array[size];
   printf("Enter %d array elements:",size);
   for(int i=0;i<size;i++)
   {
      scanf("%d",&array[i]);
   }
   int max = array[0];
   for(int i=0;i<size;i++)
   {
      if(max < array[i])
      {
         max = array[i];
      }
   }

   printf("The maximum element is %d",max);
   return 0;
}
```

4.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 size;
  printf("Enter the size: ");
  scanf("%d", &size);

  int arr[size];
  int count[size];
  int printed[size];


  for(int i = 0; i < size; i++)
  {
     count[i] = 0;
     printed[i] = 0;
  }

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

```
  }


  for(int i = 0; i < size; i++)
  {
     if (printed[i] == 0)
     {
        for(int j = 0; j < size; j++)
        {
           if(arr[i] == arr[j])
           {
              count[i]++;
           }
        }
     }


     for (int j = i; j < size; j++)
     {
        if (arr[i] == arr[j])
        {
           printed[j] = 1;
        }
     }
  }


  for(int i = 0; i < size; i++)
  {
     if(count[i] > 0 && printed[i] == 1)
     {
        printf("%d appears %d times\n", arr[i], count[i]);
     }
  }

  return 0;
}
```

ASSIGNMENT10:

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>
#define YEARS 5
#define MONTHS 12
int main()
{
```

```
    float rainfall[YEARS][MONTHS] = {
        {3.1, 2.3, 3.8, 4.0, 2.9, 1.2, 3.4, 2.5, 4.1, 3.3, 2.2, 1.8},
        {2.9, 3.0, 2.5, 4.3, 3.6, 1.5, 3.7, 2.9, 3.8, 3.2, 2.4, 2.0},
        {3.2, 2.8, 3.6, 3.9, 3.1, 1.7, 3.3, 2.6, 3.9, 3.1, 2.5, 1.9},
        {2.7, 3.4, 2.9, 4.2, 3.5, 1.3, 3.6, 2.8, 4.0, 3.4, 2.3, 1.7},
        {3.0, 2.9, 3.7, 4.1, 3.2, 1.4, 3.5, 2.7, 3.7, 3.0, 2.6, 1.5}
    };

    float totl_rainfall=0.0,yrly_totl=0.0;

    float avrg_yrly_rainfall=0.0;

    int year[5] = {2010,2011,2012,2013,2014};

    float monthlyAverage[MONTHS] = {0};

    char *months[12] =
{"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};

    printf("YEAR            RAINFALL(inches)\n");
    for (int i = 0; i < YEARS; i++)
    {
        yrly_totl = 0;
        for (int j = 0; j < MONTHS; j++)
        {
            yrly_totl += rainfall[i][j];
        }
        printf("\n%d            %.1f    \n",year[i], yrly_totl);
        totl_rainfall += yrly_totl;
    }

    avrg_yrly_rainfall = totl_rainfall/YEARS;
    printf("\nThe yearly average is %.1f inches\n",avrg_yrly_rainfall);

    printf("\nMONTHLY AVERAGES:\n");

    printf("\n\nMONTH\t\tAVERAGE RAINFALL\n");
    for (int j = 0; j < MONTHS; j++)
    {
        float monthlyTotal = 0;
        for (int i = 0; i < YEARS; i++)
        {
            monthlyTotal += rainfall[i][j];
        }
        monthlyAverage[j] = monthlyTotal / YEARS;

        printf("\n%s              %.1f \n", months[j], monthlyAverage[j]);

    }
    return 0;
}
```