Day 7

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()
{
   float const pi = 3.14;
   printf("The value of pi is %.2f\n", 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 const num = 10;
   int const *ptr = &num;
   //num = 6; Compiler error
   //*ptr = 6; Compiler error
   printf("The value of num is %d\n", num);
}
```

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 num = 10;
   int num1 =20;
   int *const ptr = &num;
    //num = 6;  //No error
    //*ptr = 6; //No error
    //ptr = &num1; //Compiler error
   printf("The value of num is %d\n", num);
}
```

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()
{
   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);
}
```

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>

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 res = add(10, 20);

   printf("%d", res);

}
```

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 days_of_week[] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday",
"Friday", "Saturday"};
```

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

    //days_of_week[0] = "NewDay"; //Compiler error
    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 the circle is: %g", area);

}
```

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 no_of_iter = 5;



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

    //no_of_iter = 10; //Compiler error

    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>
int const num = 10;
int fun(void)
{
    //num = 20; //Compiler error
    printf("In fun num = %d\n", num);
}
int main()
{
    printf("In main num = %d\n", num);
    fun();
    return 0;
}
```

ARRAYS

1.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]);
  }
}
```

2. 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

```
#include <stdio.h>
int main()
{
  int p_flag = 0, k = 2;
  int prime_arr[50] = {2, 3};

  for (int i = 4; i <= 100; i++)
  {
     p_flag = 0;
```

```
    for (int j = 2; j < i; j++)
    {
        if (i % j == 0)
        {
            p_flag = 1;
            break;
        }
    }

    if (!p_flag)
    {
        prime_arr[k] = i;
        k++;
    }
  }
  for (int i = 0; i < k; i++)
  {
    printf("%d\n", prime_arr[i]);
  }

  return 0;
}
```

3. 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;
  printf("Enter the size: ");
  scanf("%d", &size);
```

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

  for(int i=0; i<size/2; i++)
  {
     int temp=arr[i];
     arr[i]=arr[size-1-i];
     arr[size-1-i]=temp;
  }

  printf("\nReversed Array\n");
  for(int i=0; i<size; i++)
  {
     printf("%d", arr[i]);
  }
  return 0;
}
```

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

  int arr[size];
  printf("Enter the array elements: ");

  for(int i=0; i<size; i++)
  {
     scanf("%d", arr+i);
  }

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

  printf("The largest element in the array is %d\n",large);
}
```

3. 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;
}
```

4.• 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

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

```
#include <stdio.h>


int main()
{


   char *months[12] = {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov",
"Dec"};
   int years[5] = {2010, 2011, 2012, 2013, 2014};
   float rainfall[5][12] = {
      {3.3, 4.3, 4.9, 3.0, 2.3, 0.6, 1.2, 0.3, 0.5, 1.7, 3.6, 6.7}, // Total: 32.4
      {6.0, 7.3, 5.0, 4.2, 3.1, 0.5, 1.4, 0.3, 0.6, 1.3, 3.0, 5.2}, // Total: 37.5
      {8.2, 7.3, 5.5, 4.4, 3.2, 1.0, 1.1, 0.6, 0.8, 1.8, 5.0, 10.9}, // Total: 49.8
      {7.5, 6.9, 5.1, 3.8, 2.6, 0.7, 1.5, 0.3, 0.7, 1.6, 4.5, 8.8}, // Total: 44.0
      {7.5, 7.8, 4.0, 3.0, 2.3, 0.3, 1.0, 0.1, 0.4, 1.5, 1.8, 3.2}  // Total: 32.9
   };


   float yearly_total[5] = {0};
   float monthly_avg[12] = {0};
   float total_rainfall = 0;



   for (int year = 0; year < 5; year++)
   {
```

```
    yearly_total[year] = 0;
    for (int month = 0; month < 12; month++)
    {
        yearly_total[year] += rainfall[year][month];
    }
    total_rainfall += yearly_total[year];
}


for (int month = 0; month < 12; month++)
{
    for (int year = 0; year < 5; year++)
    {
        monthly_avg[month] += rainfall[year][month];
    }
    monthly_avg[month] /= 5;
}


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


printf("\nThe yearly average is %.1f inches.\n", total_rainfall / 5);


printf("\nMONTHLY AVERAGES:\n");
for (int month = 0; month < 12; month++)
```

```
    {
        printf("%s ", months[month]);
    }
    printf("\n");
    for (int month = 0; month < 12; month++)
    {
        printf("%.1f ", monthly_avg[month]);
    }

    return 0;
}
```