Assignment

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);
   //PI=3.1415;
   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()
{
   const int a=50;
    const int *ptr=&a;
   printf("%d\n",*ptr);
   //*ptr=60;

}
```

Output:

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=10;
   int b=20;
   int *const ptr=&a;
   printf("value by pointer:%d\n",*ptr);
   *ptr=30;
   printf("Modified of pointer:%d\n",*ptr);
   //ptr=&b;
   return 0;
}
```

Output:

value by pointer:10

Modified of pointer: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()
{
   const int a=10;
```

```
    const int b=20;
    const int *const ptr=&a;
    printf("%d\n",*ptr);
     *ptr=30;
     ptr=&b;
    return 0;
}
```

Output:

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>
int main()
{
    const int a=20;
    printf("%d\n",a);
    //a=50;
    return 0;
}
```

Output:

20

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 *days[]={
      "sunday","monday","tuesday","wednesday","thursday","friday","saturday"
   };
   for (int i=0;i<7;i++){
      printf("%s\n",days[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;
   const float *ptr = &PI;
```

```
    float area;
    printf("Enter the radius:");
    scanf("%d",&radius);
    area=*ptr*radius*radius;
    printf("Area of circle is:%.2f\n",area);
    return 0;
}
```

Output:

Enter the radius:2

Area of circle is:12.56

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 itr=5;
    const int *ptr=&itr;
    for(int i=0;i<=*ptr;i++){
        printf("%d\n",i+1);
    }
    return 0;
}
```

Output:

1

2

3

4

5

6

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 valu

```
#include<stdio.h>

const float PI=3.14;

int main()

{

   int radius;

   const float *ptr=&PI;

   float area;

   printf("Enter the radius:");

   scanf("%d",&radius);

   area=*ptr * radius *radius;

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

   return 0;

}
```

Output:

Enter the radius:3

Area of circle:28.260000

// •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

```
//Initializing arrays
#include<stdio.h>
int main()
{
   int primes[100]={2,3};
   int is_prime;
   int count=2;
   for(int i=4;i<=100;i++)
   {
      is_prime=1;
      for(int j=0;j<count;j++)
      {
         if(i%primes[j]==0)
         {
            is_prime=0;
            break;
         }
      }
      if(is_prime)
      {
         primes[count]=i;
         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

```
//1.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 number of elements in the array:");
    scanf("%d",&n);
    int i,arr[n],reverse[n];
    printf("Enter the elements");
    for(i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    for(int i = 0; i < n; i++) {
        reverse[i] = arr[n - i - 1];
    }
    printf("Original array:");
    for(i=0;i<n;i++){
```

```
        printf("%d",arr[i]);
    }
    printf("\n");
    printf("Reversed array:");
    for(i=0;i<n;i++)
    {
        printf("%d",reverse[i]);
    }
    printf("\n");
    return 0;
}
```

Output:

```
Enter the number of elements in the array:5
Enter the elements1
2
3
4
5
Original array:12345
Reversed array:54321
```

```
//  2. 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 the 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("Maximum element in array:%d\n",max);
}
```

Output:

```
Enter the number of elements in the array:5
Enter the elements:
6
45
96
34
55
Maximum element in array:96
```

```
// 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 n,num,count=0;
    printf("Enter the number of elements in the array:");
    scanf("%d",&n);
    int arr[n];
    printf("Enter the elements:");
    for(int i=0;i<n;i++){
        scanf("%d",&arr[i]);
    }
    printf("Enter the number for searching:");
    scanf("%d",&num);
    for(int i=0;i<n;i++){
        if(arr[i]==num){
            count++;
        }
    }
    printf("The number %d appears %d times in the array.\n",num,count);
    return 0;
}
```

Output:

```
Enter the number of elements in the array:5
Enter the elements:3
4
9
4
2
Enter the number for searching:4
The number 4 appears 2 times in the array.
```

// 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

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

```
{
   float rainfall[5][12]={
      {30.4,35.7,20.7,25.8,40.9,50.0,51.4,30.5,29.7,36,2,45.4},
      {28.6,35.7,34.0,40.2,44.6,22.9,35.8,40.6,34.7,38.5,39.5},
      {45.7,35.7,26.7,50.7,24.7,44.8,53.1,23.9,34.8,39.5,37.6},
      {23.6,24.6,53.4,39.6,28.5,32.4, 33.1, 35.6, 30.8, 31.5},
      {45.9,44.5,34.4,43.5,36.8,33.2, 32.8, 34.9, 31.2, 32.1}
   };
   float year_total,year_avg;
   float month_total,month_avg[12];
   int i,j;
   printf("Year rainfall(inches)\n");
   for(i=0;i<5;i++)
   {
      printf("%d   ",2010+i);
      year_total=0;
      for(j=0;j<12;j++)
      {
         year_total+=rainfall[i][j];
      }
      year_avg=year_total/12;
      printf("%.2f\n",year_total);
   }
   year_total=0;
   for(i=0;i<5;i++)
   {
      for(j=0;j<12;j++)
      {
         year_total+=rainfall[i][j];
      }
   }
```

```
    year_avg=year_total/(5*12);
    printf("\n The yearly average is %.2f inches.\n",year_avg);
    printf("\n Monthly Averages:\n");
    char *months[]={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
    for(j=0;j<12;j++) {
       printf("%s", months[j]);
       printf("  ");
    }
    printf("\n");
    for(j=0;j<12;j++)
    {
       month_total=0;
       for(i=0;i<5;i++)
       {
          month_total+=rainfall[i][j];
       }
       month_avg[j]=month_total/5;
       printf("%.1f",month_avg[j]);
       printf("  ");
    }
    printf("\n");
    return 0;

}
```

Output:

Year rainfall(inches)

2010 398.50

2011 395.10

2012 417.20

2013 333.10

2014 369.30

The yearly average is 31.89 inches.

Monthly Averages:

Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
34.8	35.2	33.8	40.0	35.1	36.7	41.2	33.1	32.2	35.5	15.8	9.1