DAY 7

CONST KEYWORD

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

int main()
{
    int const a=50;
    printf("001a=%d\n",a);
    a=80;
    printf("002a=%d\n",a);

    return 0;
}
```

USING POINTER

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

int main()
{
    int const a=50;//read only variable
    printf("001a=%d\n",a);
    int *p;
    p=&a;
    *p=80;


    printf("002a=%d\n",a);

    return 0;
}
```

1)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("Value of pi=%.2f",pi);

   return 0;
}
```

Output

```
Value of pi=3.14
```

2)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()
{
   float const pi=3.14;
   printf("Value of pi=%.2f\n",pi);
   float *p;
   p=(float*)&pi;
   *p=56;
   printf("Modified Value of pi=%.2f",pi);
   return 0;
}
```

Value of pi=3.14
Modified Value of pi=56.00

3)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=7,b=10;
   int *const ptr=&a;

   printf("the value of ptr is %d:",*ptr);
   ptr=&b;
   printf("the value of ptr is %d:",*ptr);

   return 0;
}
```

4)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 const a=3;
   printf("Value of a=%d\n",a);
   int *const p;
   p=(int*)&a;
   *p=56;
   printf("Modified Value of a=%d",a);
   return 0;
}
```

5)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 modify(int const a){
    a=8;
    printf("value of a=%d",a);
}
int main()
{

    modify(5);

    return 0;
}
```

6)ssignment 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()
{
    char const
arr[30][20]={"Monday","Tuesday","wednesday","Thursday","Friday","Saturday","Sunday"};

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

        printf("day %d is %s\n",i+1,arr[i]);
    }



    return 0;
}
```

7)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 pi=3.14;
   float const radius=4;
   float const area=pi*radius*radius;
   printf("Area=%.2f",area);

   return 0;
}
```

8)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()
{
   int const sum=0;
   for(int i=1;i<=6;i++)
   {
      sum=sum+i;
   }
   printf("sum:%d",sum);
   return 0;
}
```

9)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 a=6;
void fun1(){
    a=7;
    printf("a=%d\n",a);
}

void fun2(){
    printf("a=%d",a);
}
int main()
{
   fun1();
   fun2();
    return 0;
}
```

ARRAY

1)TO find address of each location

```
#include<stdio.h>
int main()
{
    int A[5];
    printf("size of int:%d\n",sizeof(int));
    printf("size of array=%d\n",sizeof(A));
    printf("address of A=%p\n",A);
    for(int i=0;i<=4;i++){
      printf("A=%p-->",(A+i));  //address of A+index*sizeof(datatype)
    }

    return 0;
}
```

OUTPUT

```
size of int:4
size of array=20
address of A=0x7fff4d1bd1c0
A=0x7fff4d1bd1c0-->A=0x7fff4d1bd1c4-->A=0x7fff4d1bd1c8-->A=0x7fff4d1bd1cc-->A=0x7fff
4d1bd1d0-->
```

2)

```
#include<stdio.h>
int main()
{
    int A[5];//MEMORY ALLOCATED FOR 5 ELEMENTS
    printf("enter the elements in the array A\n");
```

```
    for(int i=0;i<5;i++)
    {
        scanf("%d",&A[i]);
        //scanf("%d",(A+i));
        printf("\n");
    }
    for(int j=0;j<5;j++)
    {
        printf("A[%d]=%d\n",j,A[j]);
    }
    return 0;
}
```

OUTPUT

```
enter the elements in the array A
2

5

6

3

8

A[0]=2
A[1]=5
A[2]=6
A[3]=3
A[4]=8
```

3)

```
#include<stdio.h>
int main()
{
    int grades[10];
    int count=10;
    int sum=0;
    float average;
    printf("Enter the 10 grades\n");
    for(int i=0;i<count;i++)
    {
        scanf("%d",&grades[i]);
```

```
        sum=sum+grades[i];
    }
    average=(float)sum/count;
    printf("average of ten grades:%.2f",average);
    return 0;
}
```

Output

```
Enter the 10 grades
55
96
91
56
23
45
89
66
35
68
average of ten grades:62.40
```

4)

```
#include<stdio.h>
int main()
{
    int A[10]={1,2,3};
    for(int i=0;i<10;i++)
    {
        printf("A[%d]=%d\n",i,A[i]);
    }
    return 0;
}
```

```
[?2004l
A[0]=1
A[1]=2
A[2]=3
A[3]=0
A[4]=0
A[5]=0
A[6]=0
A[7]=0
A[8]=0
A[9]=0
[?2004h
```

5)DESIGNATED INTIALISERS

```
#include<stdio.h>
int main()
{
   int A[10]={[0]=45,[1]=65,[2]=76};
   for(int i=0;i<10;i++)
   {
      printf("A[%d]=%d\n",i,A[i]);
   }
   return 0;
}
```

OUTPUT

```
[?2004l
A[0]=45
A[1]=65
A[2]=76
A[3]=0
A[4]=0
A[5]=0
A[6]=0
A[7]=0
A[8]=0
A[9]=0
[?2004h
```

6)

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

[?2004l
Months 1 has 31 days
Months 2 has 28 days
Months 3 has 31 days
Months 4 has 30 days
Months 5 has 31 days
Months 6 has 30 days
Months 7 has 31 days
Months 8 has 30 days
Months 9 has 31 days
Months 10 has 31 days
Months 11 has 30 days
Months 12 has 30 days
[?2004h

7)

```
#include<stdio.h>
#define Months 12
int main()
{
    int arr[10]={0,1,4,9,16};
    for(int i=5;i<10;i++)
    {
        arr[i]=i*i;
    }
    for(int i=0;i<10;i++)
    {
        printf("%d\t",arr[i]);
    }
    return 0;
}
```

Output

0 1 4 9 16 25 36 49 64 81

1)ARRAY REVERSAL

```
#include <stdio.h>

int main() {
    int arr[20], size;
    int start, end, temp;

    printf("Enter the size of the array: ");
    scanf("%d", &size);

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

    printf("Array before reversing:\n");
    for (int i = 0; i < size; i++) {
        printf("%d\t", arr[i]);
    }


    start = 0;
    end = size - 1;


    while (start < end) {
        temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }

    printf("\nArray after reversing:\n");
    for (int i = 0; i < size; i++) {
        printf("%d\t", arr[i]);
    }

    return 0;
}
```

2)MAX VALUE

```
#include<stdio.h>
int main()
{
    int arr[20],size;
```

```
    int max=arr[0];

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

    }
    for(int i=0;i<size;i++)
    {
       printf("%d\t",arr[i]);

    }

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

    printf("\nthe maximum value is %d",max);
}
```

OUTPUT

```
[?2004l
enter size of the array:3
8
4
10
8       4       10
the maximum value is 10[?2004h
```

3)

```
#include<stdio.h>
int main()
{
    int arr[20],size,i;
    int count;

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

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

    }

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

    int count = 1;

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

        }
    }
    printf("\nthe arr[%d] appears %d times\n",i,count);
}

}
```

Output

```
[?2004l
enter size of the array:4
5
6
6
4
5       6       6       4
the arr[0] appears 1 times

the arr[1] appears 2 times

the arr[2] appears 1 times

the arr[3] appears 1 times
[?2004h
```

TWO DIMENSIONAL ARRAY

```
#include<stdio.h>
int main()
{
  int A[4][5];
  for(int i=0;i<4;i++)
  {
      for(int j=0;j<5;j++){
          printf("A[%d][%d]=%p\n",i,j,(A+i+j));
      }
  }


}
```

Output

```
[?2004l
A[0][0]=0x7ffcf7979f60
A[0][1]=0x7ffcf7979f74
A[0][2]=0x7ffcf7979f88
A[0][3]=0x7ffcf7979f9c
A[0][4]=0x7ffcf7979fb0
A[1][0]=0x7ffcf7979f74
A[1][1]=0x7ffcf7979f88
A[1][2]=0x7ffcf7979f9c
A[1][3]=0x7ffcf7979fb0
A[1][4]=0x7ffcf7979fc4
A[2][0]=0x7ffcf7979f88
A[2][1]=0x7ffcf7979f9c
A[2][2]=0x7ffcf7979fb0
A[2][3]=0x7ffcf7979fc4
A[2][4]=0x7ffcf7979fd8
A[3][0]=0x7ffcf7979f9c
A[3][1]=0x7ffcf7979fb0
A[3][2]=0x7ffcf7979fc4
A[3][3]=0x7ffcf7979fd8
A[3][4]=0x7ffcf7979fec
[?2004h
```

2)

```
#include<stdio.h>
int main()
{
  int A[4][5]={
      {1,2,3,4,5},
      {4,5,6,7},
```

```
      {11,12,13,14},
      {3,4,5,6,7}
    };
  for(int i=0;i<4;i++)
  {
      for(int j=0;j<5;j++){
          printf("A[%d][%d]=%d\n",i,j,A[i][j]);
      }
  }


}
```

```
[?2004l
A[0][0]=1
A[0][1]=2
A[0][2]=3
A[0][3]=4
A[0][4]=5
A[1][0]=4
A[1][1]=5
A[1][2]=6
A[1][3]=7
A[1][4]=0
A[2][0]=11
A[2][1]=12
A[2][2]=13
A[2][3]=14
A[2][4]=0
A[3][0]=3
A[3][1]=4
A[3][2]=5
A[3][3]=6
A[3][4]=7
[?2004h
```

3)

```
#include<stdio.h>
int main()
{
  int A[4][5]={
      {1,2,3,4},
      {4,5,6},
      {11,12,13,14},
```

```
    {3,4,5,6,7}
  };
 for(int i=0;i<4;i++)
 {
    for(int j=0;j<5;j++){
        printf("%d ",A[i][j]);
    }
    printf("\n");
 }


}
```

```
1 2 3 4 0
4 5 6 0 0
11 12 13 14 0
3 4 5 6 7
```

Assignment

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

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 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()
{
   int year=2010;
   float sum=0,average;
  float rain[5][12]={
                    {66,77,88,90,47,77,34,56,87,70,65,88},
                    {56,87,70,65,34,56,78,23,33,56,75,12},
                    {34,56,78,23,77,33,45,70,90,66,12,34},
                    {77,33,45,70,90,87,45,67,81,54,67,44},
                    {56,87,70,65,33,45,70,90,66,12,34,54}
                    };
    for(int i=0;i<5;i++){
       for(int j=0;j<12;j++){
          printf("%.2f\t",rain[i][j]);
       }
       printf("\n");
    }
    printf("\n");
    printf("YEAR\t\t\t RAINFALL(inches)");
    for(int i=0;i<5;i++){
        sum=0;
        for(int j=0;j<12;j++){
           sum =sum+rain[i][j];

        }
        average=sum/12;
        printf("\n%d\t\t\t%.2f",year,sum);
        year=year+1;
        printf("\n");
    }
    printf("\n");
    printf("Yearly average:%.2f\n",average);
    printf("\n");
    printf("Jan\t\tFeb\t\tMarch\t\tApril\t\tMay\t\tJun\t\tJuly\t\tAug\t\tSept\t\tOct\t\tNov\t\tDec");
      for(int j=0;j<12;j++){
        sum=0;
```

```
        for(int i=0;i<5;i++)
        {
           sum=sum+rain[i][j];
        }
       sum=sum/5;
       printf("%.2f\t\t\t",sum);
    }
    return 0;
}
```

[?2004l

66.00	77.00	88.00	90.00	47.00	77.00	34.00	56.00	87.00	70.00	65.00	88.00
56.00	87.00	70.00	65.00	34.00	56.00	78.00	23.00	33.00	56.00	75.00	12.00
34.00	56.00	78.00	23.00	77.00	33.00	45.00	70.00	90.00	66.00	12.00	34.00
77.00	33.00	45.00	70.00	90.00	87.00	45.00	67.00	81.00	54.00	67.00	44.00
56.00	87.00	70.00	65.00	33.00	45.00	70.00	90.00	66.00	12.00	34.00	54.00

YEAR	RAINFALL(inches)
2010	845.00
2011	645.00
2012	618.00
2013	760.00
2014	682.00

Yearly average:56.83

Jan	Feb	March	April	May	Jun
July	Aug	Sept	Oct	Nov	Dec57.80
	68.00	70.20		62.60	56.20
	59.60	54.40		61.20	71.40
	51.60	50.60		46.40	[?2004h

13.Program to print prime numbers •
In this challnge you are going to create a program that will find all the prime numbers from 3-100 •
There will 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 prime array •
You should utilize loops to only find prime numbers upto 100 and a loop to print out the prime array

```
#include <stdio.h>

int main() {
    int prime[30];
    prime[0] = 2;
    prime[1] = 3;
    int index = 2;


    for (int num = 5; num <= 100; num++) {
        int isPrime = 1;

        // Check divisibility by all previously found primes
        for (int i = 0; i < index; i++) {
            if (num % prime[i] == 0) {
                isPrime = 0;
                break;
            }
        }


        if (isPrime) {
            prime[index] = num;
            index++;
        }
    }


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

    return 0;
}
```