1.

```c
//1st example for type qualifiers......

#include <stdio.h>
const int b = 50; //Global 'const' stored in ROM or FLASH that cannot be
modified.
int main(){
    printf("\n00 %d",b);
    const int a = 50; //Local 'const' stored in RAM and can be modified
    printf("\n001 %d",a);
    //a=80; //Get o/p as -> error: assignment of read-only variable 'a'
    int *p;
    p=&a; // warning: assignment
//discards 'const' qualifier from pointer target type [-Wdiscarded-qualifiers]
    *p = 80;
    printf("\n002 %d",a);
    return 0;
}
```

2.

```c
/*
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("\nThe value for the mathematical constant Pie is %.2f",pi);
    /* pi=3.26; //error: assignment of read-only variable 'pi'
                //pi=3.26;

    printf("\nCompile Time Error : Tried to modify the constant variable
!!");*/
    return 0;
}
```

3.

```c
/*
```

```
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>
const int a = 50;
const uint8_t *p = &a;
int main()
{
    *p = 60;
    printf("The value : %d",a);
    return 0;
}
/*

After compilation
//error: assignment of read-only location '*p', *p = 60;

*/
```

4.

```
/*
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 a = 50;
int *const p = &a;
int main()
{
```

```
    // int b = 70;
    printf("Value of a: %d\n", a);
    printf("Pointer initially points to address of a with value: %d\n", *p);
    // if we try to execute this line ->  p = &b;
    //After compilation the compiler will generate this error: assignment of
read-only variable 'p'
    //   p = &b;
    *p = 60;
    printf("The value : %d",a);



    return 0;
}
```

5.

```
/*
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.

Output :error: assignment of read-only location '*p'
                *p = 60;
        error: assignment of read-only variable 'p'
                p = &b;
************************************************************/
#include <stdio.h>
int a = 50;
const int *const p = &a;
int main()
{
    // int b = 70;
    printf("Value of a: %d\n", a);
    printf("Pointer initially points to address of a with value: %d\n", *p);
    // if we try to execute this line ->  p = &b;
    //After compilation the compiler will generate this error: assignment of
read-only variable 'p'
    //   p = &b;
    *p = 60;  //error: assignment of read-only location '*p',,,,*p = 60;

    printf("The value : %d",a);
```

```
    return 0;
}
```

6.

```
/*
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.

OUTPUT : In function 'num':
c:\Users\DELL\Downloads\Program\Day7_6.c:17:6: error: assignment of read-only
parameter 'n'
     n=60;



**********************************************************************/



#include <stdio.h>

void num(const int n)
{
    n=60;
    printf("What actually happens in fn num() after the constant parameter got
changed : %d\n", n);
}
int main()
{
    int b = 70;
    printf("Value of b: %d\n", b);
    num(b);




    return 0;
```

```
}
```

7.

```
/*



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>
const char *const
arr[]={"MONDAY","TUESDAY","WEDNESDAY","THURSDAY","FRIDAY","SATURDAY","SUNDAY"}
;
int main()
{
    int l=sizeof(arr)/ sizeof(arr[0]);
    printf("\nThe Days in a week are : ");
    for(int i=0;i<l;i++)
    {
        printf("\n%s",arr[i]);
    }
    return 0;
}
```

8.

```
/*
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>
```

```c
int main()
{
    const float pi = 3.14;
    const int radius = 4;
    float area;
    area = pi * radius * radius;
    printf("\nThe value for the mathematical constant Pie is %.2f",area);

    return 0;
}
```

9.

```c
/*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 iter = 4;
    for(int i=0;i<=iter;i++)
    {
        printf("\nThis is %dth iteration.",i);
    }


    return 0;
}
```

10.

```c
/*
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 num = 4;
void num1(int x,int y){
```

```
    int sum = x+y;
    printf("\nThe sum produced by fn num1() = %d",sum);
}
void num2(int x,int y){
    int sum = x+y;
    printf("\nThe sum produced by fn num2() = %d",sum);
}
void num3(int x,int y){
    int sum = x+y;
    printf("\nThe sum produced by fn num3() = %d",sum);
}
int main()
{
    int a = 2;
    int b= 1;
    int c = 6;
    num1(num,a);
    num2(num,b);
    num3(num,c);



    return 0;
}
```

11.

```
/*Concepts of array*/
#include <stdio.h>
int main()
{
    int A[5];
    printf("\nSize of int : %d",sizeof(int));
    printf("\nSize of the array A = %d",sizeof(A));
    printf("\nA=%p",A);
    return 0;

}
```

12.

```
/*
Print the address of all locations of a array indices
*/
#include <stdio.h>
int main()
{
    int A[5];
```

```
    printf("\nSize of int : %d",sizeof(int));
    printf("\nSize of the array A = %d",sizeof(A));
    for(int i=0;i<=4;i++){
        printf("\nA=%p  -->",(A+i));//(A+i)-->Base address of A + (index value
* sizeof the datatype)
    }



    return 0;

}
```

13.

```
#include <stdio.h>
int main()
{
    int A[5];
    printf("\nEnter the elements in the array A : \n");

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


    return 0;

}
```

14.

```
#include <stdio.h>
int main(){
    int grades[10];
    int count = 10;
    long sum = 0;
    float average = 0.0f;
    printf("\nEnter the 10 grades : \n ");
    //Read the ten numbers to be averaged
    for(int i=0;i<count;++i)
    {
```

```
        printf("%2u>",i+1);

        scanf("%d",&grades[i]);
        sum+=grades[i];

    }
    average=(float)sum/count;
    printf("\nAverage of the ten grades entered is : %.2f",average);
    return 0 ;
}
```

15.

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


    return 0;
}
```

OUTPUT : [1 2 3 0 0 0 0 0 0 0]

16.

```
//Designated Initializer feature in c99 standard
#include <stdio.h>
int main()
{
    int A[10] = {[8]=90};
    printf("[");
    for(int j=0;j<10;j++)
    {
        printf(" %d",A[j]);
    }
    printf("]");


    return 0;
}
```

OUTPUT : [0 0 0 0 0 0 0 0 90 0]

17.

```c
//Example for traditional initialization
#include <stdio.h>
#define MONTHS 12
int main(void){
    int days[MONTHS]={31,28,31,30,31,30,31,31,30,31,30,31};
    int index;
    for(index=0;index<MONTHS;index++)
    {
        printf("Months %d has %2d days.\n",index+1,days[index]);
    }
    return 0;
}
```

OUTPUT :

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 31 days.

Months 9 has 30 days.

Months 10 has 31 days.

Months 11 has 30 days.

Months 12 has 31 days.

18.

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

OUTPUT :

1 31 .

2 29 .

3 0 .

4 0 .

5 31 .

6 30 .

7 31 .

8 0 .

9 0 .

10 0 .

11 0 .

12 0 .

WARNING : warning: initialized field overwritten [-Woverride-init]

int days[MONTHS]={31,28,[4]=31,30,31,[1]=29};

19.

```c
//Initializing all elmnts to the same value
int main(void){
    int array_values[10]={0,1,4,9,16};
    int i;
    for(i=5;i<10;++i){
        array_values[i]=i*i;
    }
    for(i=0;i<10;++i){
        printf("\narray_values[%i]=%i\n",i,array_values[i]);

    }
    return 0;

}
```

20.

```c
/*Task : Initializing Arrays
Requirements :
# in this challenge,u r going to create a prgm that will find the prime
numbers from 3-100
# there will be no i/p to the prgm
```

```
# the output will be each prime number seperated by a space on a single line
# u'll need to create an array that will store each prime number as it is
generated
# u can hard-code the 1st 2 prime numbers 2 &3 in the primes array
# u 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 prime_nums[100] = {2, 3};  // Array to store prime numbers, starting
with 2 and 3
    int prime_count = 2;           // Initial count of primes (since we start
with 2 primes)

    // Loop through numbers from 4 to 100
    for (int num = 4; num <= 100; num++) {
        int is_prime = 1;  // Flag to check if 'num' is prime

        // Check divisibility by all previous primes (up to sqrt(num))
        for (int i = 2; i * i <= num; i++) {
            if (num % i == 0) {
                is_prime = 0;  // Mark as not prime
                break;
            }
        }

        // If number is prime, add it to the array
        if (is_prime) {
            prime_nums[prime_count] = num;
            prime_count++;  // Increment the count of primes
        }
    }

    // Print all prime numbers in a single line
    printf("Prime numbers from 3 to 100: ");
    for (int i = 0; i < prime_count; i++) {
        printf("%d ", prime_nums[i]);
    }
    printf("\n");

    return 0;
}
```

21.

```c
/* 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(void){
    int arr[5];
    int i;
    printf("\nEnter the values inside the array : ");
    for(i=0;i<5;++i)
    {
        scanf("%d",&arr[i]);
    }
    printf("\nOriginal Array : ");
    printf("\n[");
    for(i=0;i<5;++i)
    {
        printf(" %i",arr[i]);
    }
    printf("]");

    printf("\nReversed Array : ");
    printf("\n[");
    for(i=4;i>=0;--i)
    {
        printf(" %i",arr[i]);
    }
    printf("]");
    return 0;

}
```

22.

```c
/*
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 arr[5];
    int i,max;
    printf("\nEnter the values inside the array : ");
    for(i=0;i<5;++i)
    {
        scanf("%d",&arr[i]);
```

```
    }
    printf("\nOriginal Array : ");
    printf("\n[");
    for(i=0;i<5;++i)
    {
        printf(" %i",arr[i]);
    }
    printf("]");

    for(i=0;i<5;++i)
    {
        if(arr[i]<arr[i+1])
        {
            max = arr[i+1];
        }
        else{
            max=arr[i];
        }


    }
    printf("\nThe maximum value in the array is %d",max);
    return 0;
}
```

23.

```
/*
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 arr[5];
    int i, count = 0;
    int target;

    // Input array values
    printf("Enter the values inside the array: ");
    for (i = 0; i < 5; ++i) {
        scanf("%d", &arr[i]);
    }

    // Display the original array
```

```
    printf("\nOriginal Array: [");
    for (i = 0; i < 5; ++i) {
        printf(" %d", arr[i]);
    }
    printf(" ]\n");

    // Get the target integer to search
    printf("\nEnter the integer to count in the array: ");
    scanf("%d", &target);

    // Count occurrences of the target integer
    for (i = 0; i < 5; ++i) {
        if (arr[i] == target) {
            count++;
        }
    }


    printf("\n%d is present %d times in the array.\n", target, count);

    return 0;
}
```

24.

```
//Multidimensional array Example

#include <stdio.h>

int main() {
    int A[4][5];
    for(int j=0;j<4;j++){
        for(int k =0;k<5;k++){
            printf("A[%d][%d]=%p\n",j,k,(A+j+k));
        }
    }
    return 0;
}
```

OUTPUT :

A[0][0]=0061FEC8

A[0][1]=0061FEDC

A[0][2]=0061FEF0

A[0][3]=0061FF04

A[0][4]=0061FF18

A[1][0]=0061FEDC

A[1][1]=0061FEF0

A[1][2]=0061FF04

A[1][3]=0061FF18

A[1][4]=0061FF2C

A[2][0]=0061FEF0

A[2][1]=0061FF04

A[2][2]=0061FF18

A[2][3]=0061FF2C

A[2][4]=0061FF40

A[3][0]=0061FF04

A[3][1]=0061FF18

A[3][2]=0061FF2C

A[3][3]=0061FF40

A[3][4]=0061FF54

25.

```c
//Muiltidimentional array example 2
#include <stdio.h>

int main() {
    int A[4][5]={
        {1,2,3,4,5},
        {6,7,8,9,10},
        {11,12,13,14,15},
        {16,17,18,19,20}
    };
    for(int j=0;j<4;j++){
        for(int k =0;k<5;k++){
            printf(" %d",A[j][k]);
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT :

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

26.

```
//Designated initializers in multidimensional array.
#include <stdio.h>

int main() {
    int A[4][4]={[0][0]=1,[1][1]=1,[2][2]=2,[3][3]=3};
    for(int j=0;j<4;j++){
        for(int k =0;k<4;k++){
            printf(" %d",A[j][k]);
        }
        printf("\n");
    }
    return 0;
}
```

OUTPUT :

1 0 0 0

0 1 0 0

0 0 2 0

0 0 0 3

27.

```
//Other dimensional arrays 3D
#include <stdio.h>

int main() {
    int A[2][2][2]={    //the first[2] represent the number of stack, nxt
rows,then column
        {
            {1,2},
            {3,4}
        },
        {
            {5,6},
            {7,8}
        }
    };
    int sum=0;
    for(int i=0;i<2;i++)
```

```
    {
        for(int j=0;j<2;j++)
        {
            for(int k=0;k<2;k++){
                sum+=A[i][j][k];
            }
        }
    }
    printf("\nSum of all the elements in this 3D array is %d.",sum);
    return 0;
}
```

OUTPUT :

Sum of all the elements in this 3D array is 36.

28.

```
/*
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


*/
#include <stdio.h>

int main() {
    // Declare and initialize a 2D array with rainfall data for 5 years (5
rows, 12 columns)
    float rainfall[5][12] = {
        {3.4, 2.8, 3.5, 3.9, 3.3, 2.7, 3.1, 4.0, 3.6, 2.9, 3.2, 3.0},  // 2010
        {2.9, 3.1, 4.2, 3.0, 3.4, 4.1, 3.3, 3.5, 3.9, 4.0, 2.8, 3.7},  // 2011
        {4.1, 3.6, 3.8, 4.3, 3.9, 4.4, 3.0, 3.2, 4.6, 3.7, 4.2, 4.0},  // 2012
        {3.2, 3.8, 3.9, 4.1, 3.7, 4.3, 4.0, 3.6, 3.4, 3.8, 3.9, 4.2},  // 2013
        {3.5, 2.9, 3.7, 3.0, 3.8, 3.6, 3.3, 3.9, 3.4, 3.1, 3.7, 4.0}   // 2014
```

```
    };

    int years = 5, months = 12;
    int startYear = 2010;

    // Array to store total rainfall per year
    float yearlyTotal[5] = {0};
    float totalRainfall = 0;

    // Calculate total rainfall for each year
    for (int i = 0; i < years; i++) {
        for (int j = 0; j < months; j++) {
            yearlyTotal[i] += rainfall[i][j];
        }
        totalRainfall += yearlyTotal[i];
    }

    // Calculate average yearly rainfall
    float averageYearlyRainfall = totalRainfall / years;

    // Calculate average rainfall for each month
    float monthlyAverage[12] = {0};
    for (int j = 0; j < months; j++) {
        for (int i = 0; i < years; i++) {
            monthlyAverage[j] += rainfall[i][j];
        }
        monthlyAverage[j] /= years;
    }

    // Display results
    printf("YEAR\t\tRAINFALL (inches)\n");
    for (int i = 0; i < years; i++) {
        printf("%d\t\t%.1f\n", startYear + i, yearlyTotal[i]);
    }

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

    printf("\nAverage Monthly Rainfall:\n");
    for (int j = 0; j < months; j++) {
        printf("Month %2d: %.1f inches\n", j + 1, monthlyAverage[j]);
    }

    return 0;
}
```

OUTPUT:

YEAR RAINFALL (inches)

2010	39.4
2011	41.9
2012	46.8
2013	45.9
2014	41.9

Average Yearly Rainfall: 43.2 inches

Average Monthly Rainfall:

Month 1: 3.4 inches

Month 2: 3.2 inches

Month 3: 3.8 inches

Month 4: 3.7 inches

Month 5: 3.6 inches

Month 6: 3.8 inches

Month 7: 3.3 inches

Month 8: 3.6 inches

Month 9: 3.8 inches

Month 10: 3.5 inches

Month 11: 3.6 inches

Month 12: 3.8 inches