```c
/*Write a program that declares a global variable and a local variable
with the same name.
 Modify and print both variables to demonstrate their scope and
accessibility.*/
 #include<stdio.h>
 int a=17;
 void main(){
    int b=0;
    if(b==0)
    {
        int a=0;
        printf("Value of a is %d\n",a);
    }
     printf("Value of a is %d\n",a);


 }
```

```
PS D:\projects\quest\C> cd "d:\pr
Value of a is 0
Value of a is 17
PS D:\projects\quest\C>
```

```c
/*Declare a global variable and create multiple functions to modify its
value.
 Each function should perform a different operation (e.g., addition,
subtraction) on the global variable and print its updated value.*/
 #include<stdio.h>
 int a =10;
 int add(int n, int m)
 {
    printf("Value after addition is : %d\n",n+m);
    return n+m;
 }
int subtract(int n, int m)
 {
    printf("Value after subtraction is : %d\n",n-m);
    return n-m;
 }
```

```c
 int divide(int n, int m)
 {
    printf("Value after divition is : %d\n",n/m);
    return n/m;
 }
int multiply(int n, int m)
 {
    printf("Value after multiplication is : %d\n",n*m);
    return n*m;
 }
 void main(){
a=add(a,10);
a=subtract(a,5);
a=divide(a,3);
a= multiply(a,5);
 }
```

```
PS D:\projects\quest\C> cd "d:\projects
Value after addition is : 20
Value after subtraction is : 15
Value after divition is : 5
Value after multiplication is : 25
PS D:\projects\quest\C>
```

```c
/*Write a program with a function that declares a local variable and
initializes it to a specific value.
Call the function multiple times and observe how the local variable
behaves with each call.*/
#include<stdio.h>
void  fun(void){
    int  n=10;
    printf("%d + 10 is %d\n",n,n+10);
}
void main(){
fun();
fun();
fun();
fun();
```

```
fun();
}
```

```
PS D:\projects\quest\C> cd "d:\p
10 + 10 is 20
10 + 10 is 20
10 + 10 is 20
10 + 10 is 20
10 + 10 is 20
PS D:\projects\quest\C>
```

```c
/*Write a program that calculates the sum of a global variable and a local
variable inside a function.
 Print the result and explain the variable scope in comments*/
 #include<stdio.h>
 int n=10;
 int sum(int a,int b)
{
    printf("Sum of n and m is %d \n",a+b);
    return a+b;
}
 void main(){
    int m =5;
    sum(n,m);
 }

/*
In this program  n is initialized as a global variable and has value
10,whereas the local variable m is initialized as 5,
we can invoke the variable in another  functon again and the value of n
will be the same,whereas the variable m is local to the funcion
 meaning if we invoke m outside m we will get an error
*/
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) {
Sum of n and m is 15
PS D:\projects\quest\C>
```

```c
/*Write a program that uses a global variable as a counter. Multiple
functions should increment the counter and print its value.
 Demonstrate how global variables retain their state across function
calls.*/
 #include<stdio.h>
 int counter = 0;
 void fun1()
 {
    counter++;
    printf("Value of counter : %d\n",counter);
 }
  void fun2()
 {
    counter++;
     printf("Value of counter : %d\n",counter);
 }
 void fun3()
 {
    counter++;
     printf("Value of counter : %d\n",counter);
 }
 void fun4()
 {
    counter++;
     printf("Value of counter : %d\n",counter);
 }
 void fun5()
 {
    counter++;
     printf("Value of counter : %d\n",counter);
 }
```

```
void main(){
    fun1();
    fun2();
    fun3();
    fun4();
    fun5();
}
```

```
PS D:\projects\quest\C> cd "
Value of counter : 1
Value of counter : 2
Value of counter : 3
Value of counter : 4
Value of counter : 5
PS D:\projects\quest\C>
```

```
/*Write a program where a local variable in a function shadows a global
variable with the same name.
Use the global scope operator to access the global variable and print both
values.*/
#include<stdio.h>
int n = 5;
void main(){
int n=10;
printf("Value of n is %d\n",n);
 {
extern int n;
printf("Value of n is %d\n",n);
}
}
```

```
PS D:\projects\quest\C> cd "d:\projects
Value of n is 10
Value of n is 5
PS D:\projects\quest\C>
```

```c
/*Declare a global constant variable and write a program that uses it
across multiple functions without modifying its value.
 Demonstrate the immutability of the global constant.*/
 #include<stdio.h>
 int const n =10;
 int fun1(int a ,int b)
 {
    printf("%d\n",a+b);
    return a+b;
 }
  int fun2(int a ,int b)
 {
    printf("%d\n",a-b);
    return a-b;
 }
  int fun3(int a ,int b)
 {
    printf("%d\n",a*b);
    return a*b;
 }

void main(){
printf("value of n is %d\n",n);
fun1(n,5);
fun2(n,5);
fun3(n,5);
printf("value of n is %d\n",n);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\
value of n is 10
15
5
50
value of n is 10
PS D:\projects\quest\C>
```

```c
/*Use a global variable to store configuration settings (e.g., int
configValue = 100).
 Write multiple functions that use this global configuration variable to
perform operations.*/
 #include<stdio.h>
 int cv=50;
 int fun1(int a,int b)
 {
    int r;
    r=a+b;
    printf("%d + %d = %d\n",a,b,r);
    return 0;
 }
  int fun2(int a,int b)
 {
    int r;
    r=a-b;
    printf("%d - %d = %d\n",a,b,r);
    return 0;
 }
  int fun3(int a,int b)
 {
    int r;
    r=a*b;
    printf("%d * %d = %d\n",a,b,r);
    return 0;
```

```c
}
 int fun4(int a,int b)
{
   int r;
   r=a/b;
   printf("%d / %d = %d\n",a,b,r);
   return 0;
}
void main(){
   int m =5;
   fun1(cv,m);
   fun2(cv,m);
   fun3(cv,m);
   fun4(cv,m);

}
```

```
PS D:\projects\quest\C> cd "d:\p
50 + 5 = 55
50 - 5 = 45
50 * 5 = 250
50 / 5 = 10
PS D:\projects\quest\C>
```

```c
/*Write a program where local variables are declared inside a block (e.g., if or for block).
Demonstrate that they are inaccessible outside the block.*/
#include<stdio.h>
void main(){
    for(int i= 0;i<5;i++)
    {
        printf("%d\t",i);
    }
    //printf("%d",i);
}
```

```
PS D:\projects\quest\C> cd "d:\projec
0       1       2       3       4
PS D:\projects\quest\C>
```

```c
/*Write a program that uses a global variable to track the total sum and a
local variable to store the sum of elements in an array.
Use a loop to calculate the local sum, then add it to the global total.*/
#include<stdio.h>
int total=0;
void main(){
    int n,sum,a,b;
    int ar[10]={0};
    printf("Enter the number of elements\n");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        printf("Enter the numbers\n");
        scanf("%d %d",&a,&b);
        sum=a+b;
        printf("Sum of %d and %d is  %d\n",a,b,sum);
        ar[i]=sum;
        total+=ar[i];
    }
    printf("The total sum is %d",total);
}
```

```
PS D:\projects\quest\C> cd "d:
Enter the number of elements
5
Enter the numbers
1 2
Sum of 1 and 2 is  3
Enter the numbers
2 3
Sum of 2 and 3 is  5
Enter the numbers
3 4
Sum of 3 and 4 is  7
Enter the numbers
4 5
Sum of 4 and 5 is  9
Enter the numbers
6 7
Sum of 6 and 7 is  13
The total sum is 37
PS D:\projects\quest\C>
```

```c
/*Write a program that uses a static variable inside a loop to keep track of the cumulative sum of numbers from 1 to 10.
 The loop should run multiple times, and the variable should retain its value between iterations.*/
 #include<stdio.h>
 void main(){
    static int total;
    int n,a,b,sum;
    printf("Enter the number of iterations \n");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        printf("Enter the number(between 1 and 10)\n");
        scanf("%d %d",&a,&b);
        sum=a+b;
        printf("The sum is %d\n",sum);
```

```
        total+=sum;

    }
    printf("Cumulative sum of numbers is %d",total);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\
Enter the number of iterations
5
Enter the number(between 1 and 10)
1 2
The sum is 3
Enter the number(between 1 and 10)
2 3
The sum is 5
Enter the number(between 1 and 10)
4 5
The sum is 9
Enter the number(between 1 and 10)
5 6
The sum is 11
Enter the number(between 1 and 10)
6 7
The sum is 13
Cumulative sum of numbers is 41
PS D:\projects\quest\C>
```

```
/*Use a static variable inside a loop to count the total number of
iterations
executed across multiple runs of the loop. Print the count after each
run.*/
#include<stdio.h>
void main()
{
    static int count=0;
    for(int i=0;i<10;i++)
```

```
    {for (int j = 0; j <=i; j++)
    {
        count++;

    }
     printf("Count is %d\n",count);
    }

}
```

```
PS D:\projects\quest\C> cd "
Count is 1
Count is 3
Count is 6
Count is 10
Count is 15
Count is 21
Count is 28
Count is 36
Count is 45
Count is 55
PS D:\projects\quest\C>
```

```
/*Use a static variable in a nested loop structure to count the total
number
of times the inner loop has executed across multiple runs of the
program.*/
#include<stdio.h>
void main()
{static int count=0;
    for(int i=0;i<10;i++)
    for(int j=0;j<10;j++)
    {
    count++;

    }
    printf("Value of count is %d\n",count);
}
```

```
PS D:\projects\quest\C> cd "d:\p
Value of count is 100
PS D:\projects\quest\C>
```

```c
/*Write a program where a loop executes until a specific condition is met.
Use a static variable to track and display the number of times the loop
exited due to the condition being true.*/
#include<stdio.h>
void main()
{
    for(int i=0;i<10;i++)
    for(int j=0;j<10;j++)
    {
        static int count = 0;
    if(j%5==0)
    {
        count++;
        printf("Value of count is %d\n",count);
        continue;
    }
    }
}
```

```
PS D:\projects\quest\C> cd
Value of count is 1
Value of count is 2
Value of count is 3
Value of count is 4
Value of count is 5
Value of count is 6
Value of count is 7
Value of count is 8
Value of count is 9
Value of count is 10
Value of count is 11
Value of count is 12
Value of count is 13
Value of count is 14
Value of count is 15
Value of count is 16
Value of count is 17
Value of count is 18
Value of count is 19
Value of count is 20
PS D:\projects\quest\C>
```

```c
/*Write a program where a static variable keeps track of how many times
the loop is re-entered after being interrupted (e.g., using a break
statement)*/
#include<stdio.h>
void  main(){
    static int count =0;
    for(int i=0;i<10;i++)
   {
    for(int j=0;j<10;j++)
    {
        if(j%5 ==0)
        {
        count++;
        break;
        }
    }
```

```
    }
    printf("Value of count is %d\n",count);
}
```

```
Value of count is 10
PS D:\projects\quest\C> cd "
Value of count is 10
PS D:\projects\quest\C>
```

```
/*Create a program with a loop that increments by a variable step size.
Use a static variable to count and retain the total number of steps taken
across multiple runs of the loop.*/
#include<stdio.h>
void main(){
    int a;
    printf("enter the step value\n");
    scanf("%d",&a);
    for(int i=0;i<100;i=i+a)
    {
        static int count= 0;
        count++;
        printf("Value of count is %d\n",count);
    }
}
```

```
Value of count is 7
Value of count is 8
Value of count is 9
Value of count is 10
Value of count is 11
Value of count is 12
Value of count is 13
Value of count is 14
Value of count is 15
Value of count is 16
Value of count is 17
Value of count is 18
Value of count is 19
Value of count is 20
PS D:\projects\quest\C>
```

```c
/*Declare an array of integers as const and use a loop to print each
element of the array.
Attempt to modify an element inside the loop and explain the result.*/
#include<stdio.h>
void main()
{
    const int ar[10]={1,2,3,4,5,6,7,8,9,10};
    for(int i=0;i<10;i++)
    {
      printf("%d \t",ar[i]);
        ar[i] = i;
    }

}
/* assignment of read-only location 'ar[i]'
         ar[i] = i;
         since the array contains constant values they cannot be updated
at any point of time */
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) {
122.c: In function 'main':
122.c:10:15: error: assignment of read-only location 'ar[i]'
         ar[i] = i;
               ^
PS D:\projects\quest\C>
```

```c
/*Declare a const integer variable as the upper limit of a loop.
Write a loop that runs from 0 to the value of the const variable and
prints the iteration count.*/
#include<stdio.h>
void main(){
    const int n=10;
    int count=0;
    for(int i=0;i<=n;i++)
    {
        count++;
        printf("iteration count is %d\n",count);
    }
}
```

```
PS D:\projects\quest\C> cd "d:
iteration count is 1
iteration count is 2
iteration count is 3
iteration count is 4
iteration count is 5
iteration count is 6
iteration count is 7
iteration count is 8
iteration count is 9
iteration count is 10
iteration count is 11
PS D:\projects\quest\C>
```

```c
/*Use two const variables to define the limits of nested loops.
Demonstrate how the values of the constants affect the total number of
iterations.*/
```

```c
#include<stdio.h>
void main(){
 const int n=5,m=10;
 int count =0;
 for(int i=0;i<n;i++)
     for(int j=0;j<m;j++)
        {count++;
        }
printf("Total number of iterations is %d",count);
}
```

```
PS D:\projects\quest\C> cd "d:\projects
Total number of iterations is 50
PS D:\projects\quest\C>
```

```c
/*Declare a const pointer to an integer and use it in a loop to traverse
an array.
Print each value the pointer points to.*/
#include<stdio.h>
void main(){
    int  i= 0,j;
    int ar[10]={10,9,8,7,6,5,4,3,2,1};
    int *const ptr =  &i;
    for(*ptr;*ptr<10;(*ptr)++)
    {
        j=ar[*ptr];

        printf("%d\n",j);
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects
Total number of iterations is 50
PS D:\projects\quest\C>
```

```c
/*Declare a const variable that holds a mathematical constant (e.g., PI =
3.14).
Use this constant in a loop to calculate and print the areas of circles
for a range of radii.*/
#include<stdio.h>
void main()
```

```
{
    const float pi=3.14;
    int u,l;
    int area;
    printf("Enter the range of radii\n");
    scanf("%d %d",&l,&u);
    for(int i=l;i<=u;i++)
    {
        area=pi*i*i;
        printf("Area of circle is %d\n",area);
    }
}
```

```
PS D:\projects\quest\C> cd "d
Enter the range of radii
1 5
Area of circle is 3
Area of circle is 12
Area of circle is 28
Area of circle is 50
Area of circle is 78
PS D:\projects\quest\C> cd "d
```

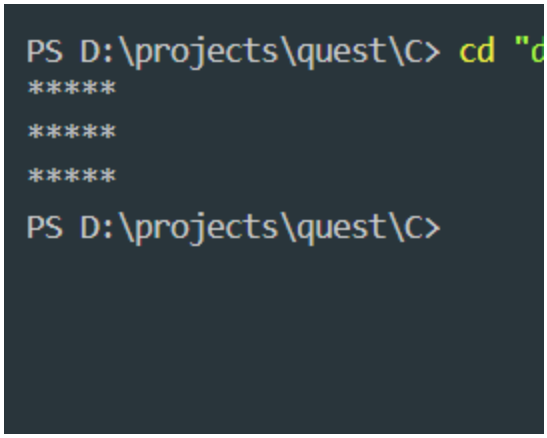
```
/*Use a const variable as a termination condition for a while loop.
The loop should terminate when the iteration count reaches the value of
the const variable*/
#include<stdio.h>
void main(){
    const int t=5;
    int count=0;
    for(int i=0;i<10;i++)
    {
        count++;
        printf("%d\n",count);
        if(count == t)
        break;
    }
}
```

```
PS D:\projects\quest\C> cd "d
1
2
3
4
5
PS D:\projects\quest\C>
```

```c
/*Declare a const variable as the step size of a for loop.
Use this step size to iterate through a range of numbers and print only
every nth number.*/
#include<stdio.h>
void main(){
    const int step=2;
    for(int i=0;i<=50;i+=step)
    {
        printf("%d \t",i);
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc 128.c -o 128 } ; if ($?) { .\128 }
0       2       4       6       8       10      12      14      16      18      20      22      24      26      28      30      32      34      36      38      40      4
2       44      46      48      50
PS D:\projects\quest\C>
```

```c
/*Use two const variables to define the number of rows and columns for
printing a rectangular pattern using nested loops.
The dimensions of the rectangle should be based on the const variables*/
#include<stdio.h>
void main()
{
    const int l=5,b=3;
    for(int j=0;j<b;j++)
    {
         for(int i=0;i<l;i++)
        printf("*");
        printf("\n");
    }
}
```


PS D:\projects\quest\C> cd "d
*****
*****
*****
PS D:\projects\quest\C>