Day 5 programs

1.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 num = 10;


int main() {
   int num = 20;


   printf("Local variable: %d\n", num);



   printf("Global variable : %d\n", num);


   return 0;
}
```

Output

```
Local variable:20
Global variable:20
```

2.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 num = 10;

void add() {
    num += 5;
    printf("After addition, num = %d\n", num);
}

void subtract() {
    num -= 3;
    printf("After subtraction, num = %d\n", num);
}


int main() {
    printf("Initial value of num = %d\n", num);

    add();
    subtract();

    return 0;
}
```

Output

```
-------
Initial value of num=100
After addition,num=110
After substraction,num=95
```

3.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 add() {
    int num=50;
    num += 10;
    printf("After addition, num = %d\n", num);
}




int main() {

    add();
    add();
    add();
    add();

    return 0;
}
```

Output

```
-------
After addition,num=60
After addition,num=60
After addition,num=60
After addition,num=60
```

4.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.

```c
#include <stdio.h>

int num1=10;
void sum() {
    int num2=50;
    int sum=num1+num2;
    printf("Sum = %d\n",sum);
}



int main() {

    sum();

    return 0;
}
```

Output

```
Sum=60
```

5.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 num=10;
void increment() {
    num=++num;
    printf("Num = %d\n",num);
}
void increment1() {
    num=++num;
    printf("Num = %d\n",num);
}
void increment2() {
    num=++num;
    printf("Num = %d\n",num);
}



int main() {

    increment();
    increment1();
    increment2();

    return 0;
}
```

Output

```
Num=11

Num=12

Num=13
```

6.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;

   {

      extern int n;

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

   }

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

}
```

Output

```
--------

Value of n is 5

Value of n is 10
```

7.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 num=10;

void increment() {
    int result=num+5;
    printf("Num = %d\n",result);
}
void increment1() {
    int result=num+6;
    printf("Num = %d\n",result);
}
void increment2() {
    int result=num+7;
    printf("Num = %d\n",result);
}


int main() {

    increment();
    increment1();
    increment2();

    return 0;
}
```

Output

```
-------
Num=15
Num=16
Num=17
```

8.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 configValue=100;
void add() {
   int result=configValue+5;
   printf("Sum = %d\n",result);
}
void multiply() {
   int result=configValue*2;
   printf("Product = %d\n",result);
}



int main() {


   add();
   multiply();


   return 0;
}
```

Output

```
Sum=105
Product=200
```

9.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>


int main() {
    int x = 10;


    if (x > 5) {
        int y = 20;
        printf("Inside the if block,y = %d\n", y);
    }


    printf("Outside the if block, x = %d\n", x);


    return 0;
}
```

Output

```
Inside the if block,y=20
Outside the if block,x=10
```

10.Problem Statement: 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 totalSum = 0;
int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    int localSum = 0;

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

    totalSum += localSum;

    printf("Sum of array elements: %d\n", localSum);
    printf("Total sum: %d\n", totalSum);

    return 0;
}
```

Output

```
Sum of array elements:15
Total sum:15
```

Storage Class Problems

1.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 CumulativeSum() {
   static int cumulativeSum = 0;

   for (int i = 1; i <= 10; i++) {
      cumulativeSum += i;
   }

   printf("Cumulative sum: %d\n", cumulativeSum);
}

int main() {
   CumulativeSum();
   CumulativeSum();

   return 0;
}
```

Output

```
--------
Cumulative Sum:55
Cumulative Sum:110
```

2.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 count() {
    static int count = 0;

    for (int i = 1; i <= 10; i++) {
        count++;
    }

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

int main() {
    count();
    count();

    return 0;
}
```

```
Output
-------
Count:10
Count:20
```

3.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 count() {
    static int count = 0;

    for (int i = 0; i<1; i++) {
        for(int j=0;j<1;j++){
            count++;
        }
    }

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

int main() {
    count();
    count();

    return 0;
}
```

Output

```
-------
Count:1
Count:2
```

4.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 count() {
    static int count = 0,num;

    while(1){
        printf("Enter a number/press 0 to stop:");
        scanf("%d",&num);

        if(num==0){
            count++;
            break;
        }
    }

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

int main() {
    count();
    count();

    return 0;
}
```

Output

```
-------
Enter a number/press 0 to stop:2
Enter a number/press 0 to stop:0
Count:1
```

5.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 count() {
   static int count = 0;


   for (int i = 1; i <= 5; i++) {
      printf("Iteration %d\n", i);


      if (i == 3) {
         count++;
         break;
      }
   }


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


int main() {
   count();
   count();


   return 0;
}
```

Output

```
-------
Iteration:1
```

```
Iteration:2
Iteration:3
Count:1
Iteration:1
Iteration:2
Iteration:3
Count:2
```

6.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.

```c
#include <stdio.h>

void countSteps() {
    static int totalSteps = 0;
    int stepSize = 2;
    int steps = 0;

    for (int i = 0; i < 10; i += stepSize) {
        steps++;
        totalSteps++;
    }

    printf("Steps in this run: %d\n", steps);
    printf("Total steps so far: %d\n", totalSteps);
}

int main() {
    countSteps();
    countSteps();
    return 0;
}
```

Output

```
Steps in this run:5
Total steps so far:5
Steps in this run:5
Total steps so far:10
```

const type specifier

1.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>


int main() {
    const int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);


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


        // Attempt to modify the element (this will cause a compile-time error)
        // arr[i] = arr[i] + 1;  // Uncommenting this line will result in an error
    }
    return 0;
}
```

Output

```
-------
Element 0:1
Element 1:2
Element 2:3
Element 3:4
Element 4:5
```

2.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

```c
#include <stdio.h>

int main() {
    int count=0;
     const int upperlimit=10;

    for(int i=0;i<=upperlimit;i++){
        count++;
    }
    printf("Count=%d\n",count);
    return 0;
}
```

Output

```
-------
Count=11
```

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

```
#include <stdio.h>


int main() {
    int count=0;
     const int num1=4;
     const int num2=3;


    for(int i=0;i<=num1;i++){
        for(int j=0;j<=num2;j++){
            count++;
        }


    }
    printf("Total no of iterations=%d\n",count);
    return 0;
}
```

Output

```
-------
Total no of iterations=20
```

4.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>


int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int size = sizeof(arr) / sizeof(arr[0]);


    const int *ptr = arr;


    for (int i = 0; i < size; i++) {
        printf("Value at index %d: %d\n", i, *ptr);


        ptr++;
    }


    return 0;
}
```

5.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>


int main() {
    const float PI = 3.14;
    float radius, area;


    for (radius = 1; radius <= 2; radius++) {
        area = PI * radius * radius;


        printf("Radius: %.2f, Area: %.2f\n", radius, area);
    }
    return 0;
}
```

Output

```
-------
Radius:1.00,Area:3.14
Radius:2.00,Area:12.56
```

6.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>

int main() {
    const int terminationValue = 5;
    int count = 0;

    while (count < terminationValue) {
        printf("Iteration count: %d\n", count);
        count++;
    }

    printf("Loop terminated after %d iterations.\n", count);

    return 0;
}
```

Output

```
-------
Iteration count:0
Iteration count:1
Iteration count:2
Iteration count:3
Iteration count:5
Loop terminated after 5 iterations
```

7.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>

int main() {
   const int stepSize = 3;
   const int rangeLimit = 6;

   for (int i = 0; i <= rangeLimit; i += stepSize) {
      printf("Current number: %d\n", i);
   }

   return 0;
}
```

8.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>
```

```
int main() {
    const int rows = 5;
    const int columns = 8;

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

        for (int j = 0; j < columns; j++) {
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```