EXPERIMENT-4

1.AIM: Declare a global variable outside all functions and use it inside various functions to understand its accessibility.

CODE:

```c
#include <stdio.h>
int a=9,b=10,c;
int sum()
{
c=a+b;
return c;
}
int multiply()
{
c=a*b;
return c;
}

int main()
{
int c1=sum();
int c2=multiply();
printf(" sum = %d\n",c1);
printf("multiplication=%d\n",c2);
return 0;
}
```

2.AIM: Declare a local variable inside a function and try to access it outside the function. Compare this with accessing the global variable from within the function

CODE:

```c
#include <stdio.h>
int a=10,b=9;
int sum()
{
int c=a+b;
printf("sum is %d", c);
return c;
}
int main()
{
sum();//c is not accessible here-ERROR:use of undeclared identifier
printf("sum=%d",c);
}
```

3.AIM: Declare variables within different code blocks(enclosed by curly braces) and test their accessibility within and outside those blocks.

CODE:

```c
#include <stdio.h>
int main()
{
 {
   int blockVar = 200;
   printf("Inside first block: blockVar = %d\n", blockVar);
 }
 /* printf("%d", blockVar); // Error: blockVar not accessible here*/

{
 int anotherVar = 300;
 printf("Inside second block: anotherVar =%d\n",anotherVar);
}
return 0;
}
```

4.AIM: Declare a static local variable inside a function . Observe how its value persists across function calls.

CODE:

```
 #include <stdio.h>
void counter()
{
static int count = 0; // Static local variable
count++;
printf("Function called %d times\n", count);
}

int main()
{
counter();
counter();
counter();

return 0;
}
```