EXPERIMENT-6

Aim : To write a Go program that displays a triangular pattern of asterisks (*) using nested loops with only two variables, i and j.

Program :

```
package main

import "fmt"

func main() {
    for i := 1; i <= 5; i++ {
        for j := 1; j <= i; j++ {
            fmt.Print("*")
        }
        fmt.Println()
    }
}
```

Output :

```
Active code page: 65001

C:\Go Programming>go run "c:\Go Programming\Q-6.go"
*
**
***
****
*****

C:\Go Programming>
```

Result : Program executed successfully.

EXPERIMENT-7

Aim : To write a Go program that multiplies two matrices using two-dimensional arrays.

Program :

```
package main

import "fmt"

func main() {
   matrix1 := [][]int{
      {1, 2, 3},
      {4, 5, 6},
      {7, 8, 9},
   }
   matrix2 := [][]int{
      {9, 8, 7},
      {6, 5, 4},
      {3, 2, 1},
   }
   result := make([][]int, len(matrix1))
   for i := range result {
      result[i] = make([]int, len(matrix2[0]))
   }
   for i := 0; i < len(matrix1); i++ {
      for j := 0; j < len(matrix2[0]); j++ {
         for k := 0; k < len(matrix2); k++ {
            result[i][j] += matrix1[i][k] * matrix2[k][j]
         }
      }
   }
   fmt.Println("Resulting matrix:")
   for _, row := range result {
      fmt.Println(row)
   }
}
```

Output :

```
C:\Go Programming>go run "c:\Go Programming\Q-7.go"
Resulting matrix:
[30 24 18]
[84 69 54]
[138 114 90]

C:\Go Programming>go run "c:\Go Programming\Q-7.go"
Resulting matrix:
[30 24 18]
[84 69 54]
[138 114 90]

C:\Go Programming>
```

Result : Program executed successfully.

EXPERIMENT-8

Aim : To write a Go program that demonstrates the use of jump statements.

Program :

```
package main

import "fmt"

func main() {
   fmt.Println("Demonstrating break:")
   for i := 0; i < 10; i++ {
      if i == 5 {
         break
      }
      fmt.Println(i)
   }

   fmt.Println("\nDemonstrating continue:")
   for i := 0; i < 10; i++ {
      if i%2 == 0 {
         continue
      }
      fmt.Println(i)
   }

   fmt.Println("\nDemonstrating return:")
   fmt.Println("Sum:", addNumbers(2, 3))
}

func addNumbers(a int, b int) int {
   return a + b
}
```

Output :

```
Demonstrating break:
0
1
2
3
4

Demonstrating continue:
1
3
5
7
9

Demonstrating return:
Sum: 5
```

Result : Program executed successfully.

EXPERIMENT-9

Aim :

Program :

```
package main

import "fmt"

func findMinMax(nums []int) (min, max int) {
   min, max = nums[0], nums[0]
   for _, num := range nums {
      if num < min {
         min = num
      }
      if num > max {
         max = num
      }
   }
   return
}

func main() {
   var n int
   fmt.Print("Enter number of elements: ")
   fmt.Scan(&n)

   nums := make([]int, n)
   fmt.Println("Enter numbers:")
   for i := range nums {
      fmt.Scan(&nums[i])
   }

   min, max := findMinMax(nums)
   fmt.Printf("Min: %d\nMax: %d\n", min, max)
}
```

Output / Input :

```
C:\Go Programming>go run "c:\Go Programming\Q-9.go"
Enter number of elements: 4
Enter numbers:
12 45 63 88
Min: 12
Max: 88

C:\Go Programming>
```

Result : Program executed successfully.

EXPERIMENT-10

Aim : To write a Go program that calculates nCr using a recursive function.

Program :

```
package main

import "fmt"

func nCr(n, r int) int {
   if r == 0 || r == n {
      return 1
   }
   return nCr(n-1, r-1) + nCr(n-1, r)
}

func main() {
   var n, r int
   fmt.Print("Enter n: ")
   fmt.Scan(&n)
   fmt.Print("Enter r: ")
   fmt.Scan(&r)
   fmt.Printf("nCr(%d, %d) = %d\n", n, r, nCr(n, r))
}
```

Output / Input :

```
C:\Go Programming>go run "c:\Go Programming\Q-10.go"
Enter n: 4
Enter r: 2
nCr(4, 2) = 6

C:\Go Programming>
```

Result : Program executed successfully.

EXPERIMENT-11

Aim : To write a Go program that swaps two numbers using a function that returns two values.

Program :

```
package main

import "fmt"

func swap(a, b int) (int, int) {
   return b, a
}

func main() {
   var x, y int
   fmt.Print("Enter the first number: ")
   fmt.Scan(&x)
   fmt.Print("Enter the second number: ")
   fmt.Scan(&y)

   fmt.Printf("Before swapping: x = %d, y = %d\n", x, y)
   x, y = swap(x, y)
   fmt.Printf("After swapping: x = %d, y = %d\n", x, y)
}
```

Output / Input :

```
C:\Go Programming>go run "c:\Go Programming\Q-11.go"
Enter the first number: 15
Enter the second number: 28
Before swapping: x = 15, y = 28
After swapping: x = 28, y = 15

C:\Go Programming>
```

Result : Program executed successfully.

EXPERIMENT-12

Aim : To write a Go program that demonstrates the basic operations with slices.

Program :

```
package main

import "fmt"

func main() {

    numbers := []int{10, 20, 30, 40, 50}


    fmt.Println("Original slice:", numbers)


    numbers = append(numbers, 60)
    fmt.Println("Slice after appending 60:", numbers)


    subSlice := numbers[1:4]
    fmt.Println("Sub-slice (1:4):", subSlice)


    numbers[2] = 35
    fmt.Println("Slice after modifying an element:", numbers)


    fmt.Println("Length of the slice:", len(numbers))
    fmt.Println("Capacity of the slice:", cap(numbers))
}
```

Output :

```
C:\Go Programming>go run "c:\Go Programming\tempCodeRunnerFile.go"
Original slice: [10 20 30 40 50]
Slice after appending 60: [10 20 30 40 50 60]
Sub-slice (1:4): [20 30 40]
Slice after modifying an element: [10 20 35 40 50 60]
Length of the slice: 6
Capacity of the slice: 10

C:\Go Programming>
```

Result : Program executed successfully.

EXPERIMENT-13

Aim : To write a Go program that performs basic string operations.

Program :

```
package main

import (
   "fmt"
   "strings"
)

func main() {
   str1 := "Hello"
   str2 := "World"

   concat := str1 + " " + str2
   fmt.Println("Concatenated string:", concat)

   length := len(concat)
   fmt.Println("Length of the concatenated string:", length)

   substr := concat[0:5]
   fmt.Println("Substring (0:5):", substr)

   upper := strings.ToUpper(concat)
   fmt.Println("Uppercase string:", upper)

   lower := strings.ToLower(concat)
   fmt.Println("Lowercase string:", lower)
}
```

Output :

```
C:\Go Programming>go run "c:\Go Programming\Q-13.go"
Concatenated string: Hello World
Length of the concatenated string: 11
Substring (0:5): Hello
Uppercase string: HELLO WORLD
Lowercase string: hello world

C:\Go Programming>
```

Result : Program executed successfully.

EXPERIMENT-14

Aim : To write a Go program that demonstrates the use of structs to create and manage an employee payroll system.

Program :

```
package main

import "fmt"

type Employee struct {
   Name   string
   ID     int
   Salary float64
}

func displayEmployee(emp Employee) {
   fmt.Printf("Employee Name: %s\n", emp.Name)
   fmt.Printf("Employee ID: %d\n", emp.ID)
   fmt.Printf("Employee Salary: %.2f\n", emp.Salary)
}

func main() {
   emp1 := Employee{Name: "Aashutosh Kumar", ID: 47, Salary: 50000.00}
   displayEmployee(emp1)
}
```

Output :

```
C:\Go Programming>go run "c:\Go Programming\Q-14.go"
Employee Name: Aashutosh Kumar
Employee ID: 47
Employee Salary: 50000.00

C:\Go Programming>
```

Result : Program executed successfully.

EXPERIMENT-15

Aim : To write a Go program that demonstrates the basic operations with maps.

Program :

```
package main

import "fmt"

func main() {
    ages := map[string]int{
        "Aashutosh": 19,
        "Parth":     25,
        "Sanjay":    21,
    }

    ages["Aashu"] = 20
    fmt.Println("Aashutosh's age:", ages["Aashutosh"])
    delete(ages, "Parth")
    fmt.Println("Updated map:", ages)

    if age, exists := ages["Sanjay"]; exists {
        fmt.Println("Sanjay's age:", age)
    } else {
        fmt.Println("Sanjay is not in the map.")
    }
}
```

Output :

```
C:\Go Programming>go run "c:\Go Programming\Q-15.go"
Aashutosh's age: 19
Updated map: map[Aashu:20 Aashutosh:19 Sanjay:21]
Sanjay's age: 21

C:\Go Programming>
```

Result : Program executed successfully.

EXPERIMENT-16

Aim : To write a Go program that demonstrates the basic usage of pointers.

Program :

```
package main

import "fmt"

func main() {
   var a int = 42
   var p *int = &a

   fmt.Println("Value of a:", a)
   fmt.Println("Address of a:", p)
   fmt.Println("Value pointed to by p:", *p)

   *p = 100
   fmt.Println("New value of a:", a)
}
```

Output :

```
C:\Go Programming>go run "c:\Go Programming\Q-16.go"
Value of a: 42
Address of a: 0xc00000a0e8
Value pointed to by p: 42
New value of a: 100

C:\Go Programming>
```

Result : Program executed successfully.

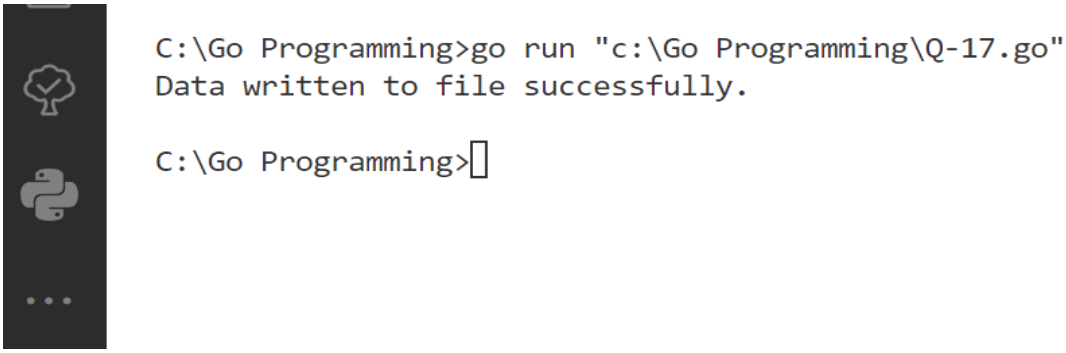
EXPERIMENT-17

Aim : To write a Go program that demonstrates how to write data into a file.

Program :

```
package main

import (
   "fmt"
   "os"
)

func main() {
   file, err := os.Create("example.txt")
   if err != nil {
      fmt.Println("Error creating file:", err)
      return
   }
   defer file.Close()

   data := "Hello, Aashutosh"
   _, err = file.WriteString(data)
   if err != nil {
      fmt.Println("Error writing to file:", err)
      return
   }

   fmt.Println("Data written to file successfully.")
}
```

Output :



Result : Program executed successfully.

EXPERIMENT-18

Aim : To write a Go program that demonstrates how to read data from a file.

Program :

```
package main

import (
   "fmt"
   "io/ioutil"
   "os"
)

func main() {
   file, err := os.Open("example.txt")
   if err != nil {
      fmt.Println("Error opening file:", err)
      return
   }
   defer file.Close()

   data, err := ioutil.ReadAll(file)
   if err != nil {
      fmt.Println("Error reading file:", err)
      return
   }

   fmt.Println("File content:")
   fmt.Println(string(data))
}
```

Output :

```
C:\Go Programming>go run "c:\Go Programming\Q-18.go"
File content:
Hello, Aashutosh

C:\Go Programming>
```

Result : Program executed successfully.

EXPERIMENT-19

Aim : To write a Go program that demonstrates the basic usage of interfaces.

Program :

```
package main

import "fmt"

type Shape interface {
   Area() float64
}

type Rectangle struct {
   Width, Height float64
}

func (r Rectangle) Area() float64 {
   return r.Width * r.Height
}

type Circle struct {
   Radius float64
}

func (c Circle) Area() float64 {
   return 3.14 * c.Radius * c.Radius
}

func main() {
   var s Shape

   s = Rectangle{Width: 5, Height: 10}
   fmt.Println("Rectangle Area:", s.Area())

   s = Circle{Radius: 7}
   fmt.Println("Circle Area:", s.Area())
}
```

Output :

```
C:\Go Programming>go run "c:\Go Programming\.vscode\Q-19.go"
Rectangle Area: 50
Circle Area: 153.86

C:\Go Programming>
```

Result : Program executed successfully.

EXPERIMENT-20

Aim : To write a Go program that demonstrates the concept of classes using structs and methods.

Program :

```
package main

import "fmt"

type Person struct {
    Name string
    Age  int
}

func (p Person) Greet() {
    fmt.Printf("Hello, my name is %s and I am %d years old.\n", p.Name, p.Age)
}

func main() {
    person1 := Person{Name: "Alice", Age: 30}
    person1.Greet()
}
```

Output :

```
C:\Go Programming>go run "c:\Go Programming\Q-20.go"
Hello, my name is Aashutosh and I am 20 years old.

C:\Go Programming>
```

Result : Program executed successfully.