---Slip 1---

Q .Write a program in GO language to accept user choice and print answers using arithmetic operators

Ans :

```
package main

import (
	"fmt"
)

func main() {
	var num1, num2 float64
	var operator string

	// Accepting user input for numbers
	fmt.Println("Enter the first number:")
	fmt.Scanln(&num1)

	fmt.Println("Enter the second number:")
	fmt.Scanln(&num2)


	fmt.Println("Enter the operator (+, -, *, /):")
	fmt.Scanln(&operator)


	switch operator {
	case "+":
		fmt.Printf("%.2f + %.2f = %.2f\n", num1, num2, num1+num2)
	case "-":
		fmt.Printf("%.2f - %.2f = %.2f\n", num1, num2, num1-num2)
	case "*":
		fmt.Printf("%.2f * %.2f = %.2f\n", num1, num2, num1*num2)
	case "/":
		if num2 != 0 {
			fmt.Printf("%.2f / %.2f = %.2f\n", num1, num2, num1/num2)
		} else {
			fmt.Println("Error: Division by zero!")
		}
	default:
		fmt.Println("Invalid operator")
	}
}
```

---slip 2--

Q. Write a program in GO language to print Fibonacci series of n terms.

```
package main

import (
	"fmt"
)

func fibonacci(n int) {
	a, b := 0, 1
	for i := 0; i < n; i++ {
		fmt.Printf("%d ", a)
		a, b = b, a+b
	}
}

func main() {
	var n int
	fmt.Println("Enter the number of terms:")
	fmt.Scanln(&n)

	fmt.Println("Fibonacci Series:")
	fibonacci(n)
}
```

--slip 3---

Q. Write a program in the GO language using function to check whether accepts number is palindrome or not

```
package main

import (
    "fmt"
    "strconv"
)
func isPalindrome(num int) bool {
    // Convert number to string
    str := strconv.Itoa(num)
    i := 0
    j := len(str) - 1

    // Iterate until pointers meet
    for i < j {

        if str[i] != str[j] {
            return false
        }
        i++
        j--
    }
    // If all characters are equal, return true
    return true
}

func main() {
    var num int
    fmt.Print("Enter a number: ")
    fmt.Scanln(&num)

    if isPalindrome(num) {
        fmt.Println(num, "is a palindrome.")
    } else {
        fmt.Println(num, "is not a palindrome.")
    }
}
```

---slip 4---

Q. Write a program in GO language to print a recursive sum of digits of a given number.

```
package main

import (
    "fmt"
)

func recursiveSumOfDigits(num int) int {

    if num < 10 {
        return num
    }
    // Recursive case: sum the last digit with the sum of digits of the remaining number
    return num%10 + recursiveSumOfDigits(num/10)
}

func main() {
    var num int
    fmt.Print("Enter a number: ")
    fmt.Scanln(&num)

    sum := recursiveSumOfDigits(num)
    fmt.Println("Recursive Sum of Digits:", sum)
}
```

--slip 5--

Q. Write a program in GO language program to create Text file

```
 package main
import (
        "fmt"
        "os"
)

func main() {

        file, err := os.Create("example.txt")
        if err != nil {
                fmt.Println("Error:", err)
                return
        }
        defer file.Close()
        text := "This is a text file created using Go programming language."
        _, err = file.WriteString(text)
        if err != nil {
                fmt.Println("Error:", err)
                return
        }

        fmt.Println("Text file created successfully.")
}
```

---slip 6--

Q. Write a program in GO language to accept two matrices and display its multiplication

```
package main

import (
    "fmt"
)

func main() {
    fmt.Println("Enter the dimensions of the first matrix (m x n):")
    fmt.Print("m: ")
    fmt.Scanln(&m)
    fmt.Print("n: ")
    fmt.Scanln(&n)
    fmt.Println("Enter the dimensions of the second matrix (p x q):")
    fmt.Print("p: ")
    fmt.Scanln(&p)
    fmt.Print("q: ")
    fmt.Scanln(&q)
    if n != p {
        fmt.Println("Matrix multiplication is not possible. The number of columns in the first matrix must be equal to the number of rows in the second matrix.")
        return
    }

    fmt.Println("Enter elements of the first matrix:")
    firstMatrix := make([][]int, m)
    for i := 0; i < m; i++ {
        firstMatrix[i] = make([]int, n)
        for j := 0; j < n; j++ {
            fmt.Printf("Enter element [%d][%d]: ", i, j)
            fmt.Scanln(&firstMatrix[i][j])
        }
    }

    fmt.Println("Enter elements of the second matrix:")
    secondMatrix := make([][]int, p)
    for i := 0; i < p; i++ {
        secondMatrix[i] = make([]int, q)
        for j := 0; j < q; j++ {
            fmt.Printf("Enter element [%d][%d]: ", i, j)
            fmt.Scanln(&secondMatrix[i][j])
        }
    }
    resultMatrix := make([][]int, m)
    for i := range resultMatrix {
        resultMatrix[i] = make([]int, q)
    }
    for i := 0; i < m; i++ {
        for j := 0; j < q; j++ {
            sum := 0
            for k := 0; k < n; k++ {
                sum += firstMatrix[i][k] * secondMatrix[k][j]
            }
            resultMatrix[i][j] = sum
        }
    }

    fmt.Println("Result of matrix multiplication:")
    for i := 0; i < m; i++ {
        for j := 0; j < q; j++ {
            fmt.Printf("%d\t", resultMatrix[i][j])
        }
        fmt.Println()
    }
}
```

---slip 7---

Q. Write a program in GO language to accept one matrix and display its transpose

```
  package main

import "fmt"

func main() {
    var m, n int
    fmt.Println("Enter the dimensions of the matrix (m x n):")
    fmt.Print("m: ")
    fmt.Scanln(&m)
    fmt.Print("n: ")
    fmt.Scanln(&n)
    fmt.Println("Enter elements of the matrix:")
    matrix := make([][]int, m)
    for i := 0; i < m; i++ {
        matrix[i] = make([]int, n)
        for j := 0; j < n; j++ {
            fmt.Printf("Enter element [%d][%d]: ", i, j)
            fmt.Scanln(&matrix[i][j])
        }
    }
    fmt.Println("Original matrix:")
    for i := 0; i < m; i++ {
        for j := 0; j < n; j++ {
            fmt.Printf("%d\t", matrix[i][j])
        }
        fmt.Println()
    }
    fmt.Println("Transpose of the matrix:")
    for j := 0; j < n; j++ {
        for i := 0; i < m; i++ {
            fmt.Printf("%d\t", matrix[i][j])
        }
        fmt.Println()
    }
}
```

--slip 8---

Q. Write a program in GO language to accept the book details such as BookID, Title, Author, Price. Read and display the details of ‘n’ number of books

```
 package main

import "fmt"
type Book struct {
    BookID int
    Title  string
    Author string
    Price  float64
}

func main() {
    var n int

    fmt.Print("Enter the number of books: ")
    fmt.Scanln(&n)

    books := make([]Book, n)


    for i := 0; i < n; i++ {
        fmt.Printf("Enter details for Book %d:\n", i+1)
        fmt.Print("BookID: ")
        fmt.Scanln(&books[i].BookID)
        fmt.Print("Title: ")
        fmt.Scanln(&books[i].Title)
        fmt.Print("Author: ")
        fmt.Scanln(&books[i].Author)
        fmt.Print("Price: ")
        fmt.Scanln(&books[i].Price)
    }
    fmt.Println("\nBook Details:")
    for i := 0; i < n; i++ {
        fmt.Printf("Book %d\n", i+1)
        fmt.Println("BookID:", books[i].BookID)
        fmt.Println("Title:", books[i].Title)
        fmt.Println("Author:", books[i].Author)
        fmt.Println("Price:", books[i].Price)
        fmt.Println()
    }
}
```

---slip 9---

Write a program in GO language using a function to check whether the accepted number is palindrome or not.

```
package main

import (
        "fmt"
        "strconv"
func isPalindrome(num int) bool {
        // Convert number to string
        str := strconv.Itoa(num)
        // Initialize two pointers
        i := 0
        j := len(str) - 1


        for i < j {
                // If characters at pointers are not equal, return false
                if str[i] != str[j] {
                        return false
                }
                i++
                j--
        }

        return true
}

func main() {
        var num int
        fmt.Print("Enter a number: ")
        fmt.Scanln(&num)


        if isPalindrome(num) {
                fmt.Println(num, "is a palindrome.")
        } else {
                fmt.Println(num, "is not a palindrome.")
        }
}
```

---slip 10---

1) Write a program in GO language to create an interface and display its values with the help of type assertion

```
 package main
import (
	"fmt"
)
type Shape interface {
	Area() float64
}

// Define a struct for Circle
type Circle struct {
	Radius float64
}

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

type Rectangle struct {
	Width  float64
	Height float64
}

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

func main() {
	shapes := []Shape{
		Circle{Radius: 5},
		Rectangle{Width: 4, Height: 3},
		Circle{Radius: 7},
	}

	for _, shape := range shapes {
		switch s := shape.(type) {
		case Circle:
			fmt.Printf("Circle - Area: %.2f\n", s.Area())
		case Rectangle:
			fmt.Printf("Rectangle - Area: %.2f\n", s.Area())
		}
	}
}
```

---slip 11---

Write a program in GO language to check whether the accepted number is two digit or not.

```
package main

import (
        "fmt"
)

func main() {
        var num int
        fmt.Print("Enter a number: ")
        fmt.Scanln(&num)
        if num >= 10 && num <= 99 {
                fmt.Println("The number", num, "is a two-digit number.")
        } else {
                fmt.Println("The number", num, "is not a two-digit number.")
        }
}
```

---slip 12--

Write a program in GO language to swap two numbers using call by reference concept

Ans

```
package main

import "fmt"

func swapByReference(a *int, b *int) {
	temp := *a
	*a = *b
	*b = temp
}

func main() {
	var num1, num2 int

	fmt.Print("Enter the first number: ")
	fmt.Scanln(&num1)
	fmt.Print("Enter the second number: ")
	fmt.Scanln(&num2)


	fmt.Println("Before swapping:")
	fmt.Println("First number:", num1)
	fmt.Println("Second number:", num2)

	swapByReference(&num1, &num2)


	fmt.Println("\nAfter swapping:")
	fmt.Println("First number:", num1)
	fmt.Println("Second number:", num2)
}
```

--slip 13--

Write a program in GO language to print sum of all even and odd numbers separately between 1 to 100.

```
package main

import "fmt"

func main() {
    sumEven := 0
    sumOdd := 0

    // Loop through numbers from 1 to 100
    for i := 1; i <= 100; i++ {
        // Check if the number is even or odd
        if i%2 == 0 {
            sumEven += i
        } else {
            sumOdd += i
        }
    }

    fmt.Println("Sum of even numbers between 1 to 100:", sumEven)
    fmt.Println("Sum of odd numbers between 1 to 100:", sumOdd)
}
```

--slip 14--

Write a program in GO language to demonstrate working of slices (like append, remove, copy etc.)

```
package main

import (
	"fmt"
)

func main() {

	slice := []int{1, 2, 3, 4, 5}
	fmt.Println("Original Slice:", slice)


	slice = append(slice, 6)
	fmt.Println("After Append:", slice)

	indexToRemove := 2
	slice = remove(slice, indexToRemove)
	fmt.Println("After Remove Element at Index", indexToRemove, ":", slice)


	copySlice := make([]int, len(slice))
	copy(copySlice, slice)
	fmt.Println("Copied Slice:", copySlice)
}

func remove(slice []int, index int) []int {
	return append(slice[:index], slice[index+1:]...)
}
```

--slip 15---

Write a program in GO language to demonstrate function return multiple values.

```
package main

import "fmt"

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

func rectangleDetails(length, width float64) (float64, float64) {
	area := length * width
	perimeter := 2 * (length + width)
	return area, perimeter
}

func main() {

	x, y := 10, 20
	fmt.Println("Before swapping:", x, y)
	x, y = swap(x, y)
	fmt.Println("After swapping:", x, y)
	length := 5.0
	width := 3.0
	area, perimeter := rectangleDetails(length, width)
	fmt.Printf("Rectangle details:\nArea: %.2f\nPerimeter: %.2f\n", area, perimeter)
}
```

--slip 16--

Write a program in GO language to create a user defined package to find out the area of a rectangle.

```
package rectangle

func Area(length, width float64) float64 {
   return length * width
}

package main

import (
   "fmt"
   "area_calculator/rectangle"
)

func main() {

   length := 5.0
   width := 3.0

   area := rectangle.Area(length, width)

   fmt.Printf("Area of the rectangle with length %.2f and width %.2f: %.2f\n", length, width, area)
}
```

---slip 17---

Write a program in GO language to illustrate the concept of returning multiple values from a function. (Add, Subtract, Multiply, Divide)

```
package main

import "fmt"
func add(a, b float64) (float64, error) {
	return a + b, nil
}
func subtract(a, b float64) (float64, error) {
	return a - b, nil
}

func multiply(a, b float64) (float64, error) {
	return a * b, nil
}

func divide(a, b float64) (float64, error) {
	if b == 0 {
		return 0, fmt.Errorf("division by zero")
	}
	return a / b, nil
}

func main() {

	num1, num2 := 10.0, 5.0


	sum, err := add(num1, num2)
	if err != nil {
		fmt.Println("Error during addition:", err)
	} else {
		fmt.Printf("Addition: %.2f + %.2f = %.2f\n", num1, num2, sum)
	}


	diff, err := subtract(num1, num2)
	if err != nil {
		fmt.Println("Error during subtraction:", err)
	} else {
		fmt.Printf("Subtraction: %.2f - %.2f = %.2f\n", num1, num2, diff)
	}
	product, err := multiply(num1, num2)
```

```
    if err != nil {
        fmt.Println("Error during multiplication:", err)
    } else {
        fmt.Printf("Multiplication: %.2f * %.2f = %.2f\n", num1, num2, product)
    }


    quotient, err := divide(num1, num2)
    if err != nil {
        fmt.Println("Error during division:", err)
    } else {
        fmt.Printf("Division: %.2f / %.2f = %.2f\n", num1, num2, quotient)
    }
}
```

--slip 18--

Write a program in GO language to print a multiplication table of number using function.

```
package main

import "fmt"

func printMultiplicationTable(num, limit int) {
    fmt.Printf("Multiplication table of %d:\n", num)
    for i := 1; i <= limit; i++ {
        fmt.Printf("%d x %d = %d\n", num, i, num*i)
    }
}

func main() {
    var number, limit int

    fmt.Print("Enter the number: ")
    fmt.Scanln(&number)
    fmt.Print("Enter the limit: ")
    fmt.Scanln(&limit)
    printMultiplicationTable(number, limit)
}
```

---slip 19---

Write a program in GO language to illustrate the function returning multiple values(add, subtract).

```
package main

import "fmt"

func add(a, b float64) float64 {
   return a + b
}

func subtract(a, b float64) float64 {
   return a - b
}

func main() {

   num1, num2 := 10.5, 5.2


   sum := add(num1, num2)
   fmt.Printf("Addition: %.2f + %.2f = %.2f\n", num1, num2, sum)


   difference := subtract(num1, num2)
   fmt.Printf("Subtraction: %.2f - %.2f = %.2f\n", num1, num2, difference)
}
```

---slip 20---

Write a program in Go language to add or append content at the end of a text file

```
package main

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

func main() {

        filePath := "example.txt"


        content := "This content is appended at the end of the file."


        err := appendToFile(filePath, content)
        if err != nil {
                fmt.Println("Error:", err)
                return
        }

        fmt.Println("Content has been successfully appended to the file.")
}

func appendToFile(filePath, content string) error {
if err != nil {
                return err
        }
        defer file.Close()

        _, err = file.WriteString(content)
        if err != nil {
                return err
        }

        return nil
}
```

===============================Iot===

a. Draw block diagram /pin diagram of Raspberry-Pi/ Beagle board /Arduino Uno board interfacing with IR Sensor/Temperature Sensor/Camera. (Internal Examiner assign any one option for board and interface device and respective interface programming option) b. WAP in python/C++ language to blink LED. c. Write down the observations on Input and Output d. Write down the Result and Conclusion

a. digram

b. WAP in Python/C++ to Blink LED (General Example):

Python (Raspberry Pi):

Python

```
import RPi.GPIO as GPIO
import time

# Set up GPIO pin (adjust based on chosen pin)
led_pin = 13

# Set GPIO mode
GPIO.setmode(GPIO.BCM)

# Set LED pin as output
GPIO.setup(led_pin, GPIO.OUT)

try:
    while True:
        # Turn LED on
        GPIO.output(led_pin, GPIO.HIGH)
        print("LED on")
        time.sleep(1)

        # Turn LED off
        GPIO.output(led_pin, GPIO.LOW)
        print("LED off")
        time.sleep(1)

except KeyboardInterrupt:
    pass

# Clean up GPIO on exit
GPIO.cleanup()
```

C++ (Arduino Uno):

C++

```
#define LED_PIN 13
```

```
void setup() {
  // Set LED pin as output
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  // Turn LED on
  digitalWrite(LED_PIN, HIGH);
  delay(1000);  // Wait for 1 second

  // Turn LED off
  digitalWrite(LED_PIN, LOW);
  delay(1000);  // Wait for 1 second
}
```

c. Observations on Input and Output:

IR Sensor Example:

- **Input:** The IR sensor detects the presence or absence of an infrared object. This results in a change in the voltage level on the connected pin (high for object detected, low for no object).
- **Output:** The program blinks the LED based on the sensor's input. LED on indicates object detected, LED off indicates no object.

d. Result and Conclusion:

- **Result:** The program successfully blinks the LED, demonstrating basic control of an output device.
- **Conclusion:** This exercise showcases a fundamental concept of single-board computers interacting with external components.

----

a. Draw block diagram /pin diagram of Raspberry-Pi/ Beagle board /Arduino Uno board interfacing with IR Sensor/Temperature Sensor/Camera. (Internal Examiner assign any one option for board and interface device and respective interface programming option) b. WAP in python/C++ language to turn ON/OFF buzzer. c. Write down the observations on Input and Output d. Write down the Result and Conclusion

b. WAP in Python/C++ Language to Turn On/Off Buzzer:

Python (Raspberry Pi):

Python

```python
import RPi.GPIO as GPIO
import time

# Set up GPIO pin (adjust based on chosen pin)
buzzer_pin = 18

# Set GPIO mode
GPIO.setmode(GPIO.BCM)

# Set buzzer pin as output
GPIO.setup(buzzer_pin, GPIO.OUT)

try:
    while True:
        # Turn buzzer on
        GPIO.output(buzzer_pin, GPIO.HIGH)
        print("Buzzer on")
        time.sleep(2)  # Adjust duration for on time

        # Turn buzzer off
        GPIO.output(buzzer_pin, GPIO.LOW)
        print("Buzzer off")
        time.sleep(1)  # Adjust duration for off time

except KeyboardInterrupt:
    pass

# Clean up GPIO on exit
GPIO.cleanup()
```

C++ (Arduino Uno):

C++

```cpp
#define BUZZER_PIN 8

void setup() {
  // Set buzzer pin as output
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop() {
  // Turn buzzer on
  digitalWrite(BUZZER_PIN, HIGH);
  delay(2000);  // Adjust duration for on time

  // Turn buzzer off
  digitalWrite(BUZZER_PIN, LOW);
  delay(1000);  // Adjust duration for off time
}
```

c. Observations on Input and Output:

- **Input:** In this example, there's no external input involved. The program controls the buzzer on/off cycle.
- **Output:** The output is the buzzing sound generated by the buzzer when the connected pin is set to HIGH. Turning the pin LOW stops the buzzer sound.

d. Result and Conclusion:

- **Result:** The program successfully controls the buzzer, turning it on and off based on the defined

----

a. Draw block diagram /pin diagram of Raspberry-Pi/ Beagle board /Arduino Uno board interfacing with IR Sensor/Temperature Sensor/Camera. (Internal Examiner assign any one option for board and interface device and respective interface programming option) b. WAP in python/C++ language to blink LED. c. Write down the observations on Input and Output d. Write down the Result and Conclusion

b. WAP in Python/C++ Language to Blink LED (General Example):

Python (Raspberry Pi):

Python

```
import RPi.GPIO as GPIO
import time

# Set up GPIO pin (adjust based on chosen pin)
led_pin = 13

# Set GPIO mode
GPIO.setmode(GPIO.BCM)

# Set LED pin as output
```

```
GPIO.setup(led_pin, GPIO.OUT)

try:
    while True:
        # Turn LED on
        GPIO.output(led_pin, GPIO.HIGH)
        print("LED on")
        time.sleep(1)

        # Turn LED off
        GPIO.output(led_pin, GPIO.LOW)
        print("LED off")
        time.sleep(1)

except KeyboardInterrupt:
    pass

# Clean up GPIO on exit
GPIO.cleanup()
```

C++ (Arduino Uno):

C++

```
#define LED_PIN 13

void setup() {
  // Set LED pin as output
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  // Turn LED on
  digitalWrite(LED_PIN, HIGH);
  delay(1000);  // Wait for 1 second

  // Turn LED off
  digitalWrite(LED_PIN, LOW);
  delay(1000);  // Wait for 1 second
}
```

c. Observations on Input and Output:

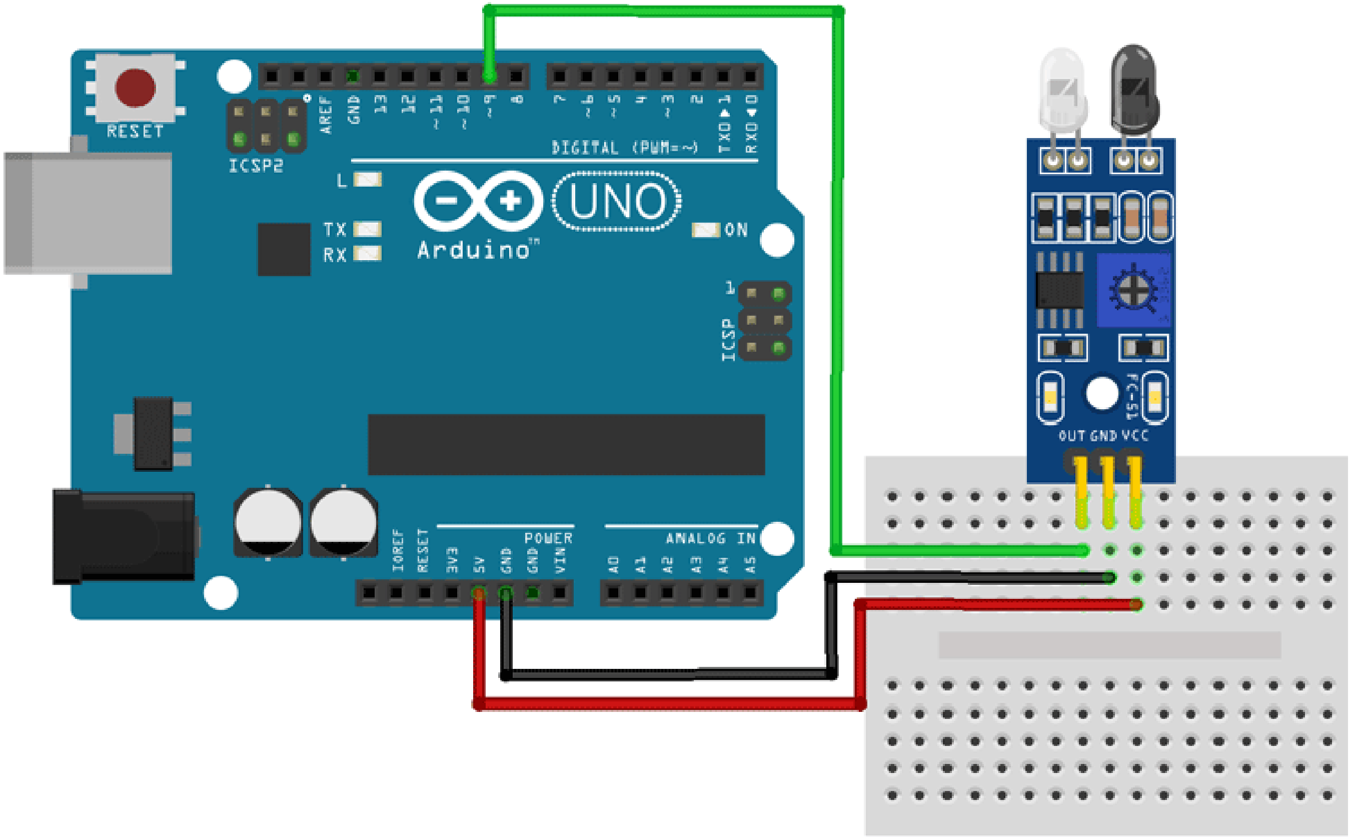
- **Input:** There's no external input in this example. The program controls the LED blinking pattern.

- **Output:** The LED turns on and off based on the program's instructions. You'll observe a blinking LED light.

d. Result and Conclusion:

- **Result:** The program successfully blinks the LED, demonstrating control of an external device.
- **Conclusion:** This exercise showcases a fundamental concept of controlling output devices with single-board computers. You can adapt the code to control other devices based on sensor readings or user input.

Digram : draw block diagram arduino uno board interfacing with ir sensor


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX▶1
RX◀0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
1
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
ANALOG IN
A0
A1
A2
A3
A4
A5
FC-51
OUT GND VCC