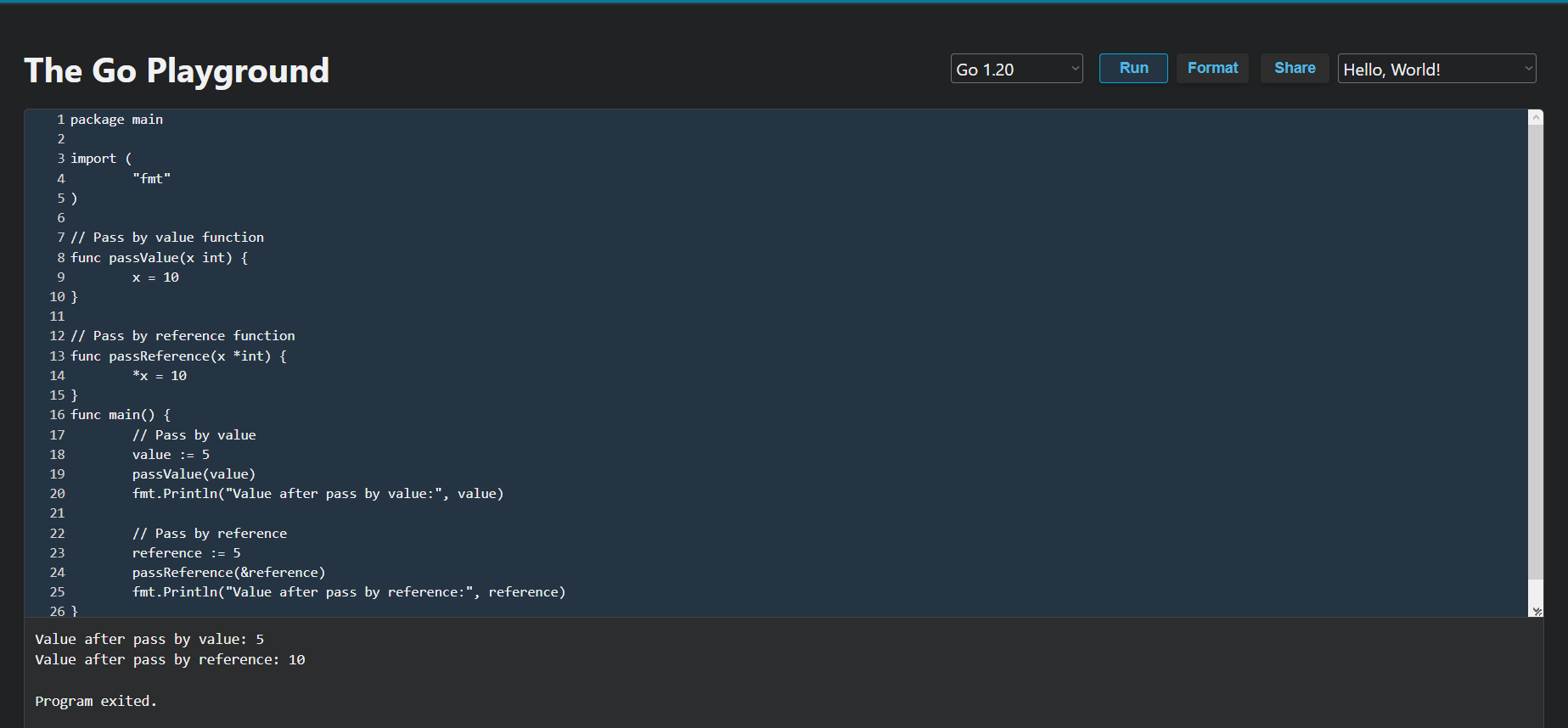
**SATHVIK ANDELA**

**20WU0101033**

Q. Create a Program to explain difference between Pass by Value and pass by Reference in Golang



package main

import (

"fmt"

)

// Pass by value function

func passValue(x int) {

x = 10

}

// Pass by reference function

func passReference(x \*int) {

\*x = 10

}

func main() {

// Pass by value

value := 5

passValue(value)

fmt.Println("Value after pass by value:", value)

// Pass by reference

reference := 5

passReference(&reference)

fmt.Println("Value after pass by reference:", reference)

}

There are two functions in this programme: passByValue and passByReference. When given an integer argument, x, the passByValue function gives it the value "10." Yet, because x is passed by value instead of by reference, the initial value of value in the main function is unaltered.

On the other hand, the passByReference function's input (\*int) is a pointer to an integer. The variable pointed to by the pointer (\*x) is then given the value 10 after the pointer has been dereferenced using the \* operator. The value of reference in the main function is changed to 10 since reference is passed by reference (i.e., as a pointer).

This demonstrates the difference between pass by value and pass by reference - when we pass a variable by value, a copy of its value is created and passed to the function, so changes made to the value inside the function don't affect the original variable. When we pass a variable by reference, we pass a pointer to the variable, so changes made to the value inside the function affect the original variable.

In summary, pass by value creates a copy of the value of a variable and any changes made inside the function do not affect the original variable, whereas pass by reference allows the function to modify the original variable directly by passing its address.