***iOS Assignment – 2***

**Ques : 1. Truth table of XOR.**

func XOR(\_ A : Bool, \_ B : Bool) -> Bool {

if (A && !B) || (!A && B) {

return true

}

return false

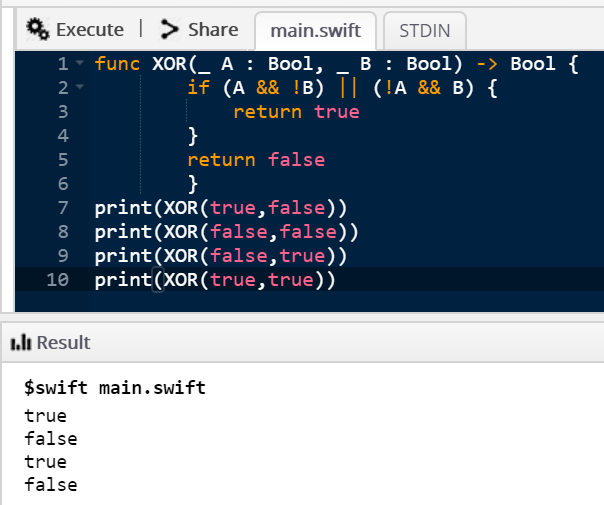
}

print(XOR(true,false))

print(XOR(false,false))

print(XOR(false,true))

print(XOR(true,true))



**Ques : 2. Reverse of a String.**

var str : String = "Jatin Vats"

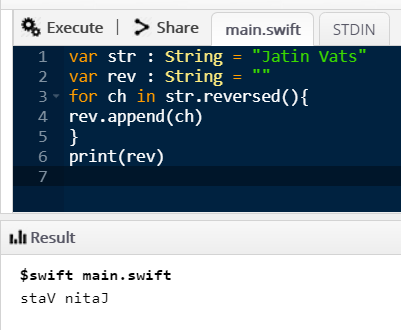
var rev : String = ""

for ch in str.reversed(){

rev.append(ch)

}

print(rev)



**Ques : 3. String is palindrome or not.**

func palindrome(str : String){

let rev = str.reversed()

if (str.elementsEqual(rev)) {

print("\(str) is a palindrome") }

else {

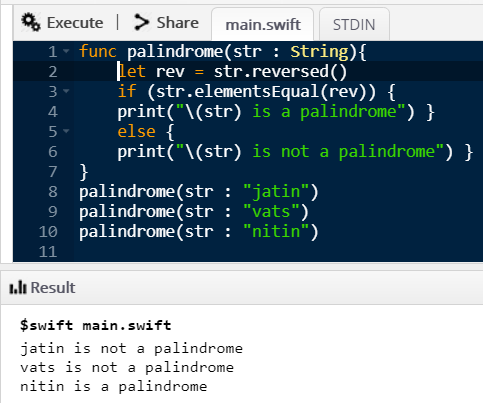
print("\(str) is not a palindrome") }

}

palindrome(str : "jatin")

palindrome(str : "vats")

palindrome(str : "nitin")



**Ques : 4. Binary Search.**

func binary(\_ arr: [Int],\_ value: Int) -> Int?

{

var left = 0

var right = arr.count - 1

while left <= right {

let middle = Int((Double(left + right) / 2.0))

if arr[middle] < value {

left = middle + 1

} else if arr[middle] > value {

right = middle - 1

} else {

return middle

}

}

return nil

}

func input(\_ arr: [Int],\_ value: Int){

if let index = binary(arr,value) {

print("Found \(value) at index \(index)")

}

else {

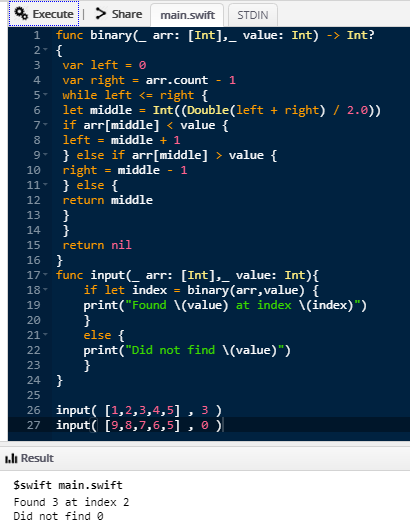
print("Did not find \(value)")

}

}

input( [1,2,3,4,5] , 3 )

input( [9,8,7,6,5] , 0 )



**Ques : 5. Function to convert integer number to full words.**

import Foundation

func con(num : Int){

let strNum = NumberFormatter()

strNum.numberStyle = NumberFormatter.Style.spellOut

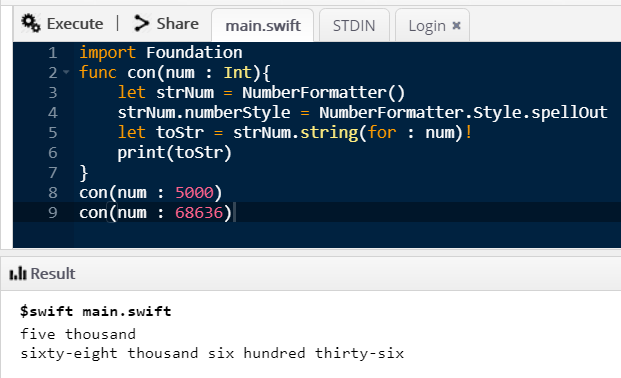
let toStr = strNum.string(for : num)!

print(toStr)

}

con(num : 5000)

con(num : 68636)



**Ques : 6. Convert a string to uppercase or lowercase.**

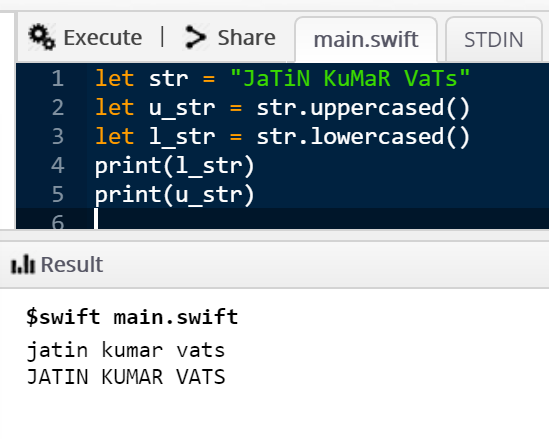
let str = "JaTiN KuMaR VaTs"

let u\_str = str.uppercased()

let l\_str = str.lowercased()

print(l\_str)

print(u\_str)



**Ques : 7. Find out the missing number in an array.**

func missing(\_ arr : [Int]){

let n = arr.count

var total = (n+1)\*(n+2)/2

for i in arr{

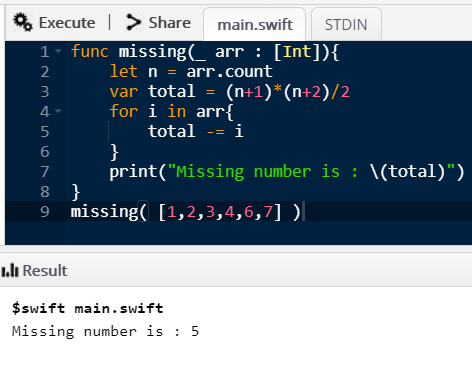
total -= i

}

print("Missing number is : \(total)")

}

missing( [1,2,3,4,6,7] )



**Ques : 8. Find out the Mean of numbers an an array.**

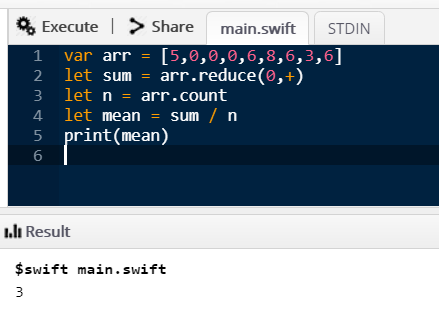
var arr = [5,0,0,0,6,8,6,3,6]

let sum = arr.reduce(0,+)

let n = arr.count

let mean = sum / n

print(mean)



**Ques : 9. Bracket Validation.**

func validate(\_ str : String){

var c = 0

for i in str{

if ( i == "(" ) {

c = c+1

}

else if ( i == ")"){

if(c==0){

print("!!!Entered bracket sequence is not valid") }

else{

c = c-1 }

}

}

if(c==0){

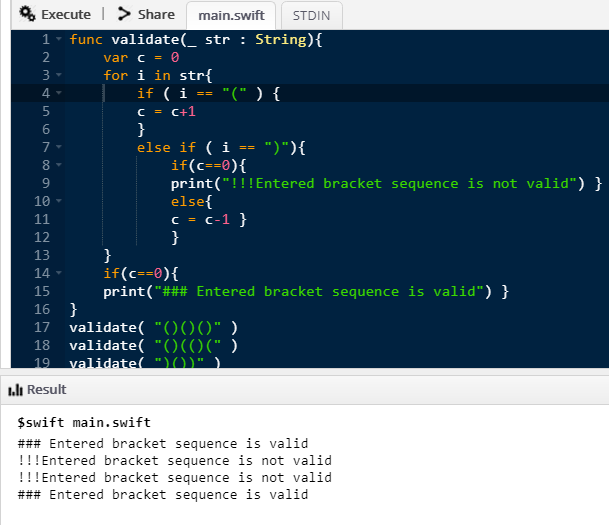
print("### Entered bracket sequence is valid") }

}

validate( "()()()" )

validate( "()(()(" )

validate( ")())" )



**Ques : 10. Pre-order and In-order Traversal**

var Pre : [Int] = []

var In : [Int] = []

class Node {

var data: Int

var left: Node?

var right: Node?

init(data: Int) {

self.data = data

}

}

func preOrderTraversal(node: Node?) {

guard let node = node else { return }

Pre.append(node.data)

//print(node.data)

preOrderTraversal(node: node.left)

preOrderTraversal(node: node.right)

}

func inOrderTraversal(node: Node?) {

guard let node = node else { return }

inOrderTraversal(node: node.left)

In.append(node.data)

//print(node.data)

inOrderTraversal(node: node.right)

}

var root = Node(data: 55)

root.left = Node(data: 40)

root.left?.left = Node(data: 15)

root.left?.right = Node(data: 27)

root.right = Node(data: 99)

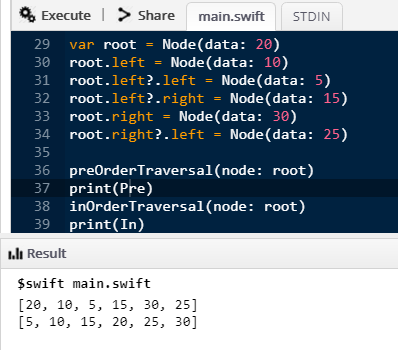
root.right?.left = Node(data: 12)

preOrderTraversal(node: root)

print(Pre)

inOrderTraversal(node: root)

print(In)



**Ques : 11. Write a program to construct completely balanced binary tree.**

class Node {

var value: String

var children: [Node] = []

weak var parent: Node?

init(value: String) {

self.value = value

}

func add(child: Node) {

children.append(child)

child.parent = self

}

}

extension Node: CustomStringConvertible {

var description: String {

var text = "\(value)"

if !children.isEmpty {

text += " {" + children.map { $0.description }.joined(separator: ", ") + "} "

}

return text

}

}

let dogs = Node(value: "Dogs")

let dog1 = Node(value: "Beagle")

let dog2 = Node(value: "Corgi")

dogs.add(child: dog1)

dogs.add(child: dog2)

print(dogs)

