func strange(\_ x : Int, \_ y : Int)-> Int

{

if (x > y)

{

return (x + y)

}

else

{

return (x - y)

}

}

print(strange(4, 5 ))

func isMult(\_ x: Int) -> Bool

{

if x % 26 == 0 {

return true

}

else {

return false

}

}

//What's displayed by the following statement?

print(isMult(5))

func addTen(n : Int)-> Int{

return n + 10

}

func addTen2(\_ n : Int)-> Int{

return n + 10

}

func celsiusToFahrenhiet(\_ c : Double) -> Double

{

var f : Double

f = 1.8 \* c + 32

return f

}

func celsiusToFahrenhiet2(c : Double) -> Double

{

var f : Double

f = 1.8 \* c + 32

return f

}

func fahrenhietToCelsius(\_ f : Double) -> Double

{

var c : Double

c = (f - 32.0) \* (5.0 / 9.0)

return c

}

func fahrenhietToCelsius2(f : Double) -> Double

{

var c : Double

c = (f - 32.0) \* (5.0 / 9.0)

return c

}

func main()

{

print(addTen(n:10))

print(addTen2(1000))

print("Temperature in Celsius:",terminator:" ")

let cons\_c = (Double)(readLine()!)

var c : Double = cons\_c!

print("Temperature in Fahrenhiet: ", celsiusToFahrenhiet(c))

print("Temperature in Fahrenhiet: ", celsiusToFahrenhiet2(c: c))

print("Temperature in Fahrenhiet: ",terminator:" ")

let cons\_f = (Double)(readLine()!)

var f : Double = cons\_f!

print("Temperature in Celsius: ", fahrenhietToCelsius(f))

print("Temperature in Celsius: ", fahrenhietToCelsius2(f: f))

}

main()

func rectangle(w: Double, l: Double) -> (Double, Double)

{

var area: Double, perimeter : Double;

area = w \* l

perimeter = 2 \* (w + l)

return (area, perimeter)

}

func Main()

{

let (area, perimeter) = rectangle(w: 5, l: 10)

print("Area: ",area)

print("Perimeter: ", perimeter)

// OR

let result = rectangle(w: 6, l: 8)

print("Area: ",result.0)

print("Perimeter: ", result.1)

}

Main()

import Foundation

let pi\_val = 3.14159265

let calories\_per\_square\_inch = 16.7

func calc\_circle\_area(pizza\_diameter: Double) -> (Double)

{

var circle\_radius, circle\_area : Double

circle\_radius = pizza\_diameter / 2.0

circle\_area = pi\_val \* circle\_radius \* circle\_radius

return circle\_area

}

func pizza\_calories(pizza\_diameter : Double) -> Double

{

var total\_calories : Double

total\_calories = calc\_circle\_area(pizza\_diameter:pizza\_diameter) \* calories\_per\_square\_inch

return total\_calories

}

func Main()

{

var calories, pizza\_diameter : Double

pizza\_diameter = 12

calories = pizza\_calories(pizza\_diameter:pizza\_diameter)

print(pizza\_diameter," inches pizza has ", String(format: "%6.2f",calories)," calories")

}

Main()

import Foundation

func payroll(hoursWorked : Double, payRate : Double) -> (Double, Double, Double, Double)

{

var regularPay : Double, overtimePay : Double = 0, overtimeHour : Double, grossPay : Double, tax : Double, netPay : Double

if hoursWorked <= 40

{

regularPay = hoursWorked \* payRate

}

else

{

regularPay = 40 \* payRate

overtimeHour = hoursWorked - 40

overtimePay = overtimeHour \* payRate \* 1.5

}

grossPay = regularPay + overtimePay

tax = grossPay \* 0.05 // tax is 5% of gross pay

netPay = grossPay - tax

return (grossPay, tax, overtimePay, netPay)

}

func Main()

{

var hoursWorked : Double, payRate : Double

let (overtimePay, grossPay, tax, netPay) = payroll(hoursWorked: 40, payRate: 10)

print(String(format: "Gross pay: %6.2f", grossPay), String(format:" Tax: %6.2f", tax), String(format:" Over Time Pay: %6.2f ", overtimePay), String(format:" NetPay: %6.2f", netPay))

hoursWorked = 42.0

payRate = 10.0

let (overtimePay2, grossPay2, tax2, netPay2) = payroll(hoursWorked: hoursWorked, payRate: payRate)

print(String(format: "Gross pay: %6.2f", grossPay2), String(format:" Tax: %6.2f", tax2), String(format:" Over Time Pay: %6.2f ", overtimePay2), String(format:" NetPay: %.2f", netPay2))

hoursWorked = 40

payRate = 10.0

let result = payroll(hoursWorked: hoursWorked, payRate: payRate)

print(String(format: "Gross pay: %6.2f", result.0), String(format:" Tax: %6.2f", result.1), String(format:" Over Time Pay: %6.2f ", result.2), String(format:" NetPay: %6.2f", result.3))

}

Main()

//Global variable - taxRate

let taxRate = 0.0825

func payroll(hoursWorked : Double, payRate : Double) -> (Double, Double)

{

var grossPay, netPay : Double

grossPay = hoursWorked \* payRate

netPay = grossPay - (grossPay \* taxRate)

return (grossPay, netPay)

}

func Main()

{

var hoursWorked, payRate : Double

print("Hours worked: ",terminator:"")

let const\_hoursWorked = (Double)(readLine()!)

hoursWorked = const\_hoursWorked!

print("Hours worked: ",terminator:"")

let const\_payRate = (Double)(readLine()!)

payRate = const\_payRate!

let (grossPay, netPay) = payroll(hoursWorked:hoursWorked, payRate:payRate)

print("Gross Pay: ", grossPay)

print("Net Pay: ", netPay)

print("Tax Rate: ",taxRate)

}

Main()

//Convet the measure from feet and inches to cm

let INCHES\_PER\_INCH = 12.0

let CM\_PER\_INCH = 2.54

func height\_US\_to\_cm(feet : Double, inches : Double) -> Double

{

//Convert height in feet / inches to centimeters

var total\_inches, cm : Double

total\_inches = feet \* INCHES\_PER\_INCH

cm = total\_inches \* CM\_PER\_INCH

return cm

}

func Main()

{

var feet , inches : Double

var centimeters : Double

feet = 6.0 ; inches = 4.0

print("Length in Feet: ", terminator:"")

let const\_feet = (Double)(readLine()!)

feet = const\_feet!

print("Length in Inches: ", terminator:"")

let const\_inches = (Double)(readLine()!)

inches = const\_inches!

centimeters = height\_US\_to\_cm(feet : feet, inches : inches)

print ("Centimeters : ", centimeters)

}

Main()