* Rewrite the provided subtraction code in standard arithmetic notation (e.g., 2 + 4).
* Print the difference using the println function.

// Calculate the difference between 8 and 5

val difference = 8-5

// Print the difference

println(difference)

* Define variables for cards 2, 3, and 4 for the suit clubs (♣). Explicitly specify their type as Int.

// Define immutable variables for clubs 2♣ through 4♣

val twoClubs: Int = 2

val threeClubs: Int = 3

val fourClubs: Int = 4

* Define playerA, playerB, and playerC as immutable variables with the strings "Alex", "Chen", and "Marta", respectively.
* Read the code that *reassigns* playerC to "Umberto". Click "Run Code" and observe the error.
* Delete the code that reassigns playerC to "Umberto".
* Change the original playerC variable definition from "Marta" to "Umberto".

// Define immutable variables for player names

val playerA: String = "Alex"

val playerB: String = "Chen"

val playerC: String = "Umberto"

// Change playerC from Marta to Umberto

//playerC = "Umberto"

Define four mutable variables, all of type Int and equal to 1: aceClubs, aceDiamonds, aceHearts, aceSpades

// Define mutable variables for all aces

var aceClubs: Int=1

var aceDiamonds: Int=1

var aceHearts: Int=1

var aceSpades: Int=1

* Create a var named playerA and with the name Alex as a string literal as its value.
* Change the point value of the ace of diamonds so it is worth 11 points, instead of the original value of 1 point it was assigned.
* Add jackClubs and aceDiamonds to calculate the value of Alex's hand and print the result.

// Create a mutable variable for Alex as player A

var playerA: String = "Alex"

// Change the point value of A♦ from 1 to 11

var aceDiamonds: Int = 11

// Calculate hand value for J♣ and A♦

println(jackClubs + aceDiamonds)

* Calculate the hand value for playerA, who has the following cards: queenDiamonds, threeClubs, aceHearts (worth 1), fiveSpades.
* Calculate the hand value for playerB, who has the following cards: kingHearts, jackHearts.
* Call the maxHand function, passing in handPlayerA and handPlayerB as arguments. Pass this function call into the println function to print out the maximum hand value.

// Calculate hand values

var handPlayerA: Int = queenDiamonds + threeClubs + aceHearts + fiveSpades

var handPlayerB: Int = kingHearts + jackHearts

// Find and print the maximum hand value

println(maxHand(handPlayerA, handPlayerB))

Create and parameterize an array (named hands) of type Int with a length of 3. Explicitly provide the type parameterization.

// Create and parameterize an array for a round of Twenty-One

var hands: Array[Int] = new Array[Int](3)

* Initialize the first player's hand in the hands array.
* Initialize the second player's hand in the hands array.
* Initialize the third player's hand in the hands array.

// Create and parameterize an array for a round of Twenty-One

val hands: Array[Int] = new Array[Int](3)

// Initialize the first player's hand in the array

hands(0) = tenClubs + fourDiamonds

// Initialize the second player's hand in the array

hands(1) = nineSpades + nineHearts

// Initialize the third player's hand in the array

hands(2) = twoClubs + threeSpades

* Create an array named hands.
* Initialize the first player's hand to tenClubs + fourDiamonds.
* Initialize the second player's hand to nineSpades + nineHearts.
* Initialize the third player's hand to twoClubs + threeSpades

// Create, parameterize, and initialize an array for a round of Twenty-One

val hands = Array(tenClubs + fourDiamonds,

nineSpades + nineHearts,

twoClubs + threeSpades)

* Add a fiveClubs to the first player's hand in hands.
* Add a queenSpades to the second player's hand in hands.
* Add a kingClubs to the third player's hand in hands.

// Initialize player's hand and print out hands before each player hits

hands(0) = tenClubs + fourDiamonds

hands(1) = nineSpades + nineHearts

hands(2) = twoClubs + threeSpades

hands.foreach(println)

// Add 5♣ to the first player's hand

hands(0) = hands(0) + fiveClubs

// Add Q♠ to the second player's hand

hands(1) = hands(1) + queenSpades

// Add K♣ to the third player's hand

hands(2) = hands(2) + kingClubs

// Print out hands after each player hits

hands.foreach(println)

* Initialize a list named prizes with an element for each round's prize, where the first through fifth round's prizes are 10, 15, 20, 25, and 30, respectively.
* Prepend to prizes using the cons (::) operator so a new first round is added, worth $5. Name the new list newPrizes

// Initialize a list with an element for each round's prize

val prizes = List(10,15,20,25,30)

println(prizes)

// Prepend to prizes to add another round and prize

val newPrizes = 5 :: prizes

println(newPrizes)

Using :: and Nil, initialize a list named prizes with an element each round's prize, where the first through fifth round's prizes are 10, 15, 20, 25, and 30, respectively.

// Initialize a list with an element each round's prize

val prizes = 10 :: 15 :: 20 :: 25 :: 30 :: Nil

println(prizes)

* Create the venuesNTOA list using List().
* Create the venuesEuroTO list using the cons operator (::).
* Concatenate venuesNTOA and venuesEuroTO to create a new list named venuesTOWorld

// The original NTOA and EuroTO venue lists

val venuesNTOA = List("The Grand Ballroom", "Atlantis Casino", "Doug's House")

val venuesEuroTO = "Five Seasons Hotel" :: "The Electric Unicorn" :: Nil

// Concatenate the North American and European venues

val venuesTOWorld = venuesNTOA ::: venuesEuroTO

* Write code that accomplishes the following: if a player's hand is equal to 21, print "Twenty-One!" to output.
* Click "Run Code" and observe the output.
* Change fourSpades to threeSpades, then click "Submit Answer".

// Point value of a player's hand

val hand = sevenClubs + kingDiamonds + threeSpades

// Congratulate the player if they have reached 21

if (hand == 21) {

println("Twenty-One!")

}

* Read the provided code, then fill in the type annotation for the informPlayer variable.
* Write appropriate if, else if, and else conditions based on the provided code.
* Change fiveSpades to fourSpades, then click "Run Code" and observe the output.
* Change fourSpades to threeSpades, then click "Submit Answer".

// Point value of a player's hand

val hand = sevenClubs + kingDiamonds + threeSpades

// Inform a player where their current hand stands

val informPlayer: String = {

if (hand > 21)

"Bust! :("

else if (hand == 21)

"Twenty-One! :)"

else

"Hit or stay?"

}

// Print the message

print(informPlayer)

* Write an if condition: if the hand busts, make the Int 0 the result of the function.
* Write an else condition: if the hand does not bust, subtract hand from 21 and make that difference the result of the function.
* Call pointsToBust with a hand with the cards tenSpades and fiveClubs as the argument.

// Find the number of points that will cause a bust

def pointsToBust(hand: Int): Int = {

// If the hand is a bust, 0 points remain

if (bust(hand))

0

// Otherwise, calculate the difference between 21 and the current hand

else

21 - hand

}

// Test pointsToBust with 10♠ and 5♣

val myHandPointsToBust = pointsToBust(tenSpades + fiveClubs)

println(myHandPointsToBust)

* Define a counter variable, i, equal to zero.
* Define the number of iterations for the while loop, numRepetitions, equal to three.
* Fill out the first if clause so *"winner"* is printed in the first iteration of the loop, then *"winner"* again in the second iteration, and "chicken dinner" in the final iteration.
* Increment the counter variable by one.

// Define counter variable

var i = 0

// Define the number of loop iterations

var numRepetitions = 3

// Loop to print a message for winner of the round

while (i < numRepetitions) {

if (i < 2)

println("winner")

else

println("chicken dinner")

// Increment the counter variable

i = i + 1

}

* Define a counter variable, i. Set it equal to zero to start.
* Using the counter variable i, write a while loop that proceeds through the loop hands.length times.
* Find and print the winning hand's value for the ith hand.
* Increment the counter variable.

// Define counter variable

var i = 0

// Create list with five hands of Twenty-One

var hands = List(16, 21, 8, 25, 4)

// Loop through hands

while (i < (hands.length)) {

// Find and print number of points to bust

println(pointsToBust(hands(i)))

// Increment the counter variable

i = i + 1

}

Call the foreach method on the hands array of arrays, looping through each round to find the number of points to bust using the pointsToBust function.

// Find the number of points that will cause a bust

def pointsToBust(hand: Int) = {

// If the hand is a bust, 0 points remain

if (bust(hand))

println(0)

// Otherwise, calculate the difference between 21 and the current hand

else

println(21 - hand)

}

// Create list with five hands of Twenty-One

var hands = List(16, 21, 8, 25, 4)

// Loop through hands, finding each hand's number of points to bust

hands.foreach(pointsToBust)