-----------------------

-- Samuil Daniela Teodora

-- 07.10.2020

-----------------------

-- Edit the lines above with your name and the submission date.

> type Face = Ace | Two | Three | Four | Five | Six | Seven | Eight | Nine | Ten | Jack | Queen | King

> Ace

Ace : Face

> Eight

Eight : Face

> Queen

Queen : Face

> type Suit = Clubs | Diamonds | Hearts | Spades

> Clubs

Clubs : Suit

> Diamonds

Diamonds : Suit

> Spades

Spades : Suit

> type Card = Card Face Suit

> Card

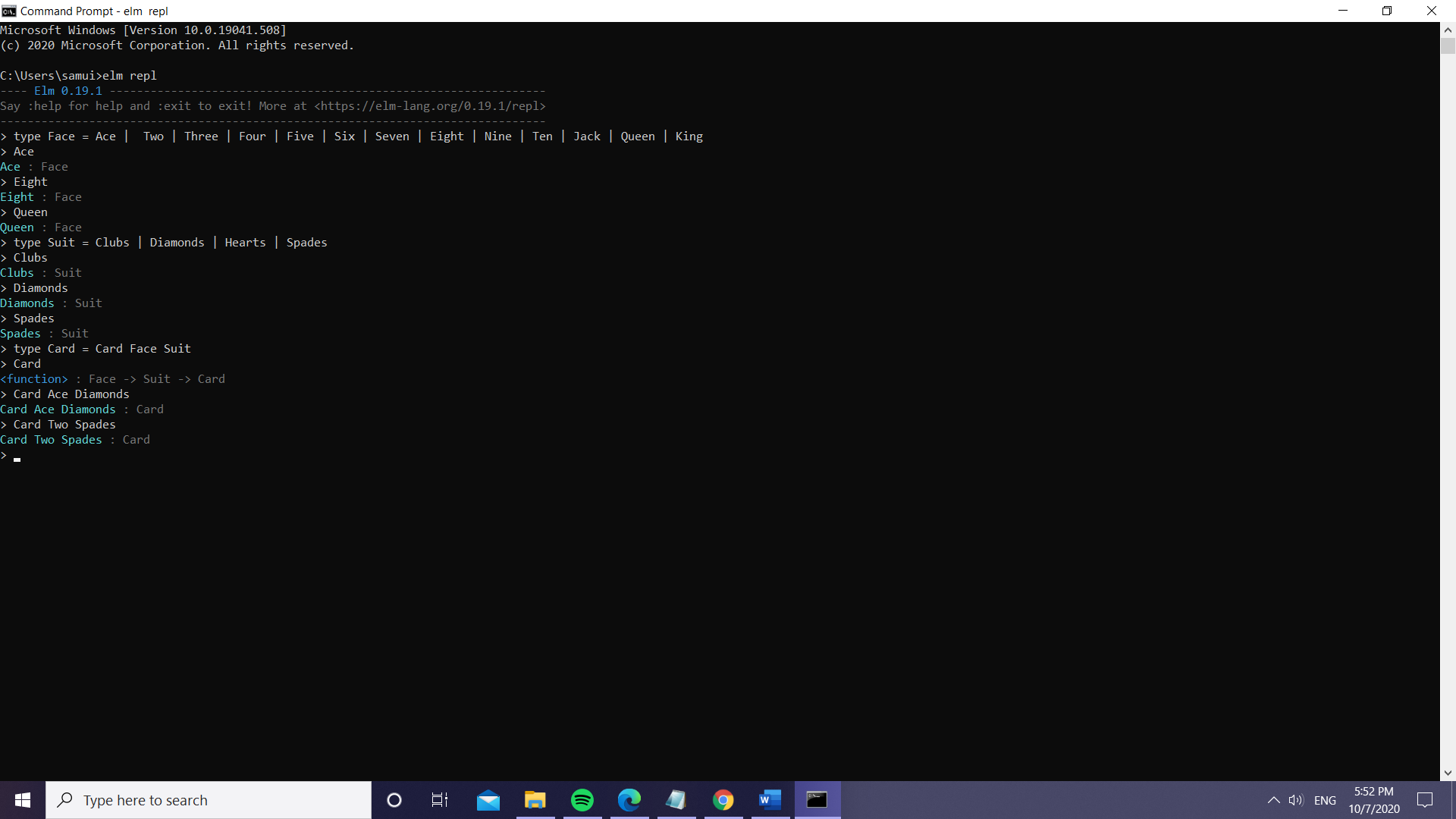
<function> : Face -> Suit -> Card

> Card Ace Diamonds

Card Ace Diamonds : Card

> Card Two Spades

Card Two Spades : Card



> faceToString : Face -> String

| faceToString face =

| case face of

| Ace -> "The face is Ace"

| Two -> "The face is Two"

| Three -> "The face is Three"

| Four -> "The face is Four"

| Five -> "The face is Five"

| Six -> "The face is Six"

| Seven -> "The face is Seven"

| Eight -> "The face is Eight"

| Nine -> "The face is Nine"

| Ten -> "The face is Ten"

| Jack -> "The face is Jack"

| Queen -> "The face is Queen"

| King -> "The face is King"

|

<function> : Face -> String

> faceToString Ace

"The face is Ace" : String

> faceToString Queen

"The face is Queen" : String

> suitToString: Suit -> String

| suitToString suit =

| case suit of

| Clubs -> "The suit is Clubs"

| Diamonds -> "The suit is Diamonds"

| Spades -> "The suit is Spades"

| Hearts -> "The suit is Hearts"

|

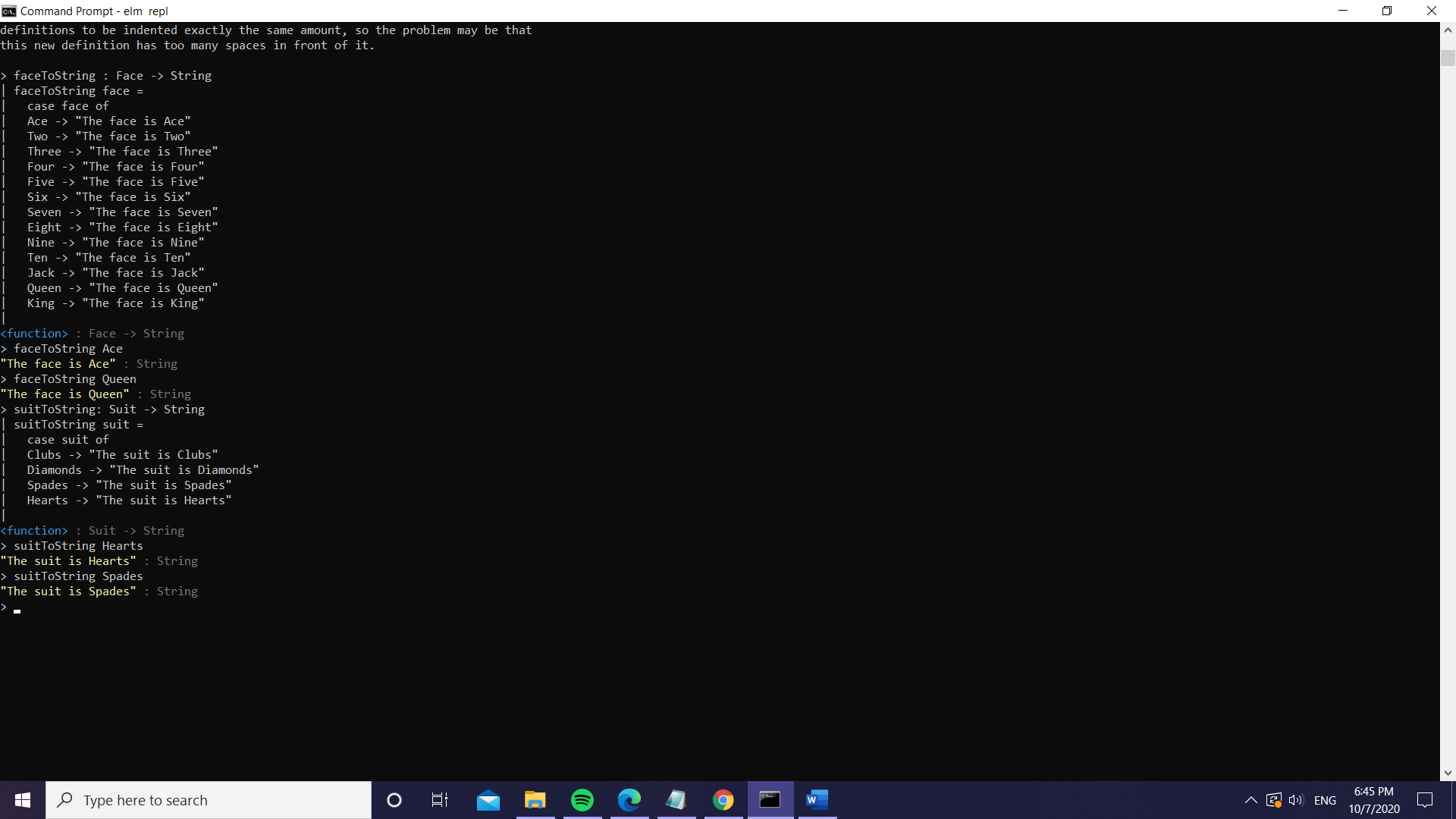
<function> : Suit -> String

> suitToString Hearts

"The suit is Hearts" : String

> suitToString Spades

"The suit is Spades" : String



cardToString: Card -> String

cardToString card =

let

(Card face suit) = card

in

case (face,suit) of

(Ace,Clubs) -> "The card is Ace of Clubs"

(Ace,Diamonds) -> "The card is Ace of Diamonds"

(Ace,Spades) -> "The card is Ace of Spades"

(Ace,Hearts) -> "The card is Ace of Hearts"

(Two,Clubs) -> "The card is Two of Clubs"

(Two,Diamonds) -> "The card is Two of Diamonds"

(Two,Spades) -> "The card is Two of Spades"

(Two,Hearts) -> "The card is Two of Hearts"

(Three,Clubs) -> "The card is Three of Clubs"

(Three,Diamonds) -> "The card is Three of Diamonds"

(Three,Spades) -> "The card is Three of Spades"

(Three,Hearts) -> "The card is Three of Hearts"

(Four,Clubs) -> "The card is Four of Clubs"

(Four,Diamonds) -> "The card is Four of Diamonds"

(Four,Spades) -> "The card is Four of Spades"

(Four,Hearts) -> "The card is Four of Hearts"

(Five,Clubs) -> "The card is Five of Clubs"

(Five,Diamonds) -> "The card is Five of Diamonds"

(Five,Spades) -> "The card is Five of Spades"

(Five,Hearts) -> "The card is Five of Hearts"

(Six,Clubs) -> "The card is Six of Clubs"

(Six,Diamonds) -> "The card is Six of Diamonds"

(Six,Spades) -> "The card is Six of Spades"

(Six,Hearts) -> "The card is Six of Hearts"

(Seven,Clubs) -> "The card is Seven of Clubs"

(Seven,Diamonds) -> "The card is Seven of Diamonds"

(Seven,Spades) -> "The card is Seven of Spades"

(Seven,Hearts) -> "The card is Seven of Hearts"

(Eight,Clubs) -> "The card is Eight of Clubs"

(Eight,Diamonds) -> "The card is Eight of Diamonds"

(Eight,Spades) -> "The card is Eight of Spades"

(Eight,Hearts) -> "The card is Eight of Hearts"

(Nine,Clubs) -> "The card is Nine of Clubs"

(Nine,Diamonds) -> "The card is Nine of Diamonds"

(Nine,Spades) -> "The card is Nine of Spades"

(Nine,Hearts) -> "The card is Nine of Hearts"

(Ten,Clubs) -> "The card is Ten of Clubs"

(Ten,Diamonds) -> "The card is Ten of Diamonds"

(Ten,Spades) -> "The card is Ten of Spades"

(Ten,Hearts) -> "The card is Ten of Hearts"

(Jack,Clubs) -> "The card is Jack of Clubs"

(Jack,Diamonds) -> "The card is Jack of Diamonds"

(Jack,Spades) -> "The card is Jack of Spades"

(Jack,Hearts) -> "The card is Jack of Hearts"

(Queen,Clubs) -> "The card is Queen of Clubs"

(Queen,Diamonds) -> "The card is Queen of Diamonds"

(Queen,Spades) -> "The card is Queen of Spades"

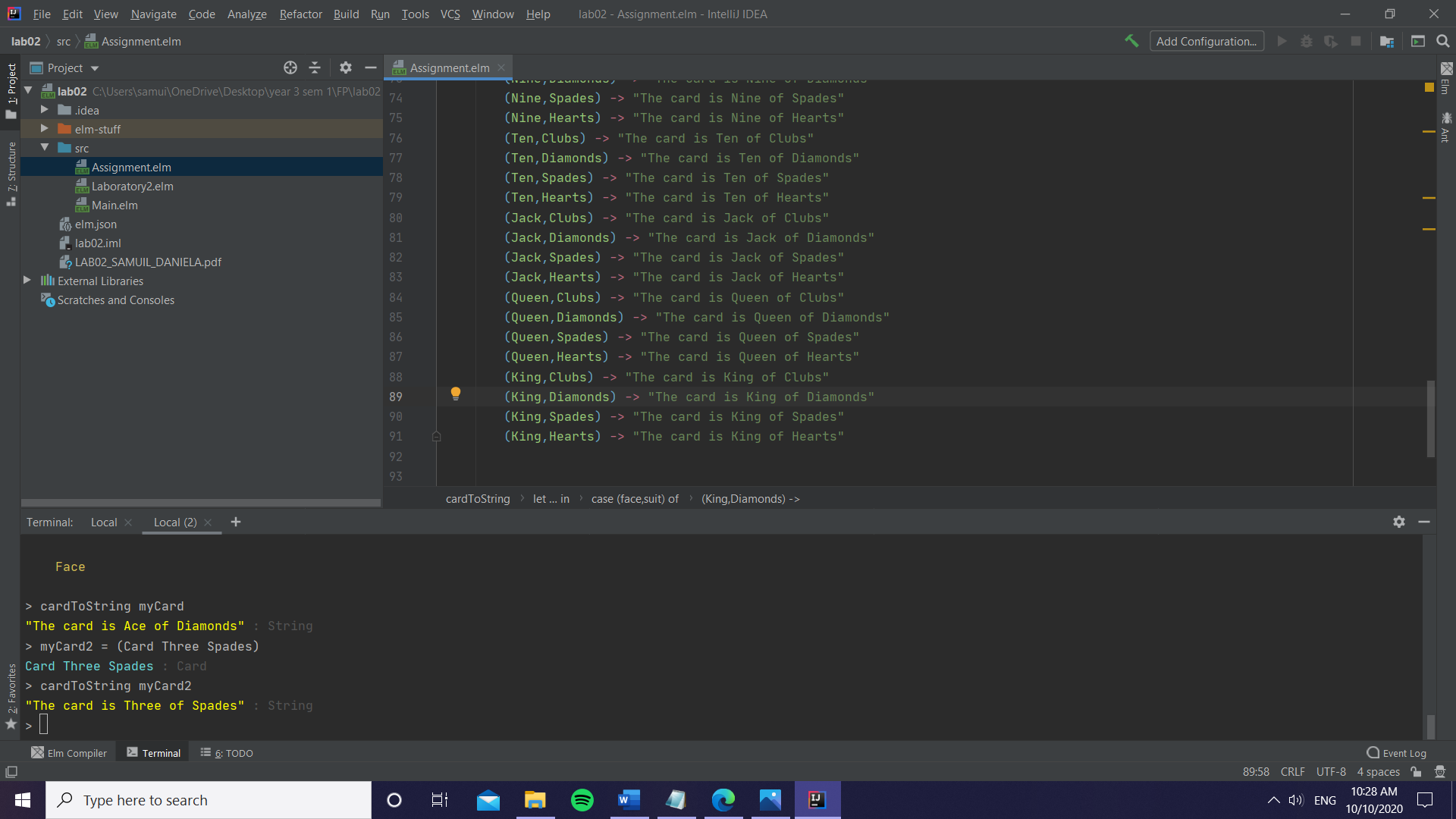
(Queen,Hearts) -> "The card is Queen of Hearts"

(King,Clubs) -> "The card is King of Clubs"

(King,Diamonds) -> "The card is King of Diamonds"

(King,Spades) -> "The card is King of Spades"

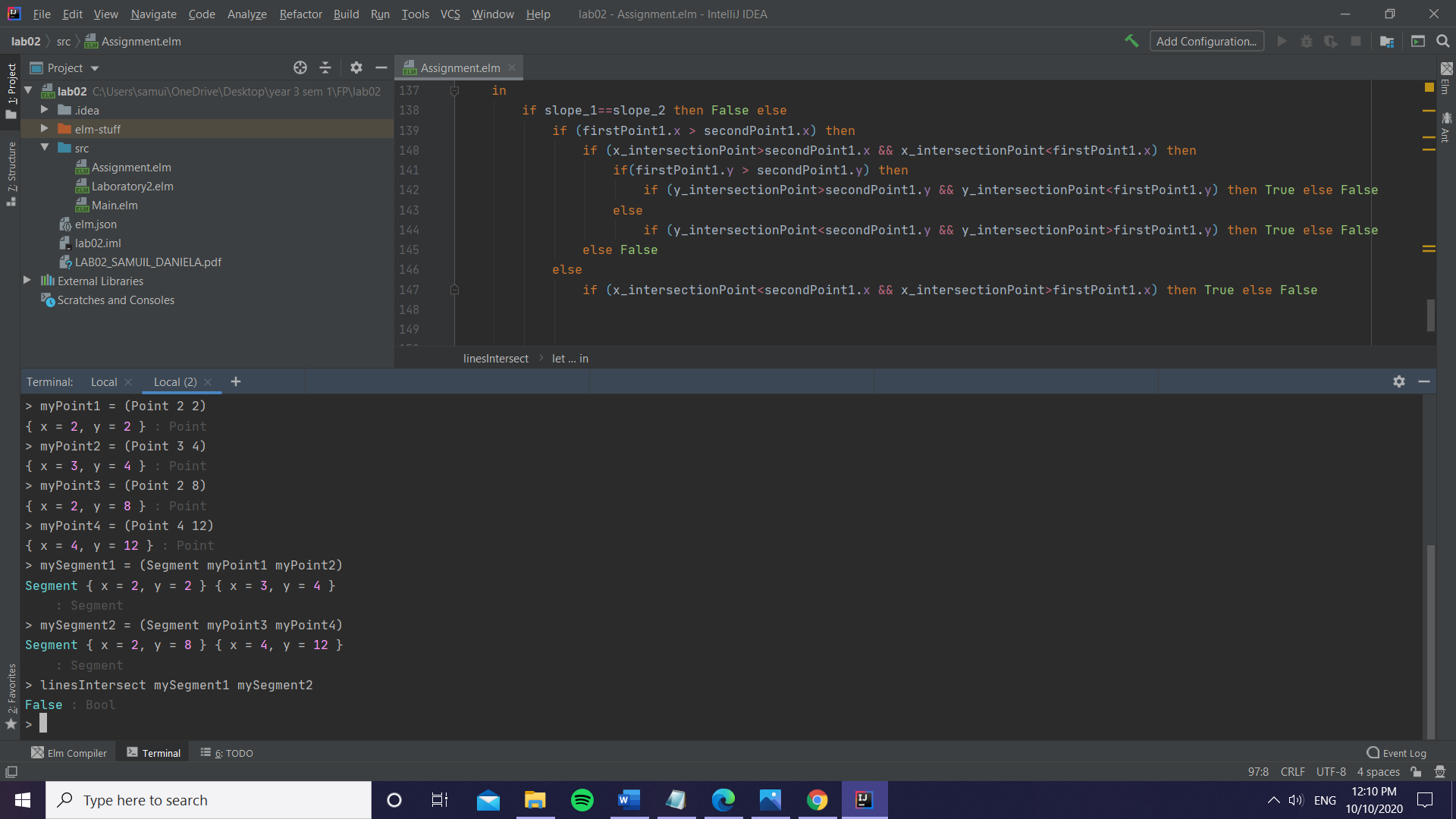
(King,Hearts) -> "The card is King of Hearts"

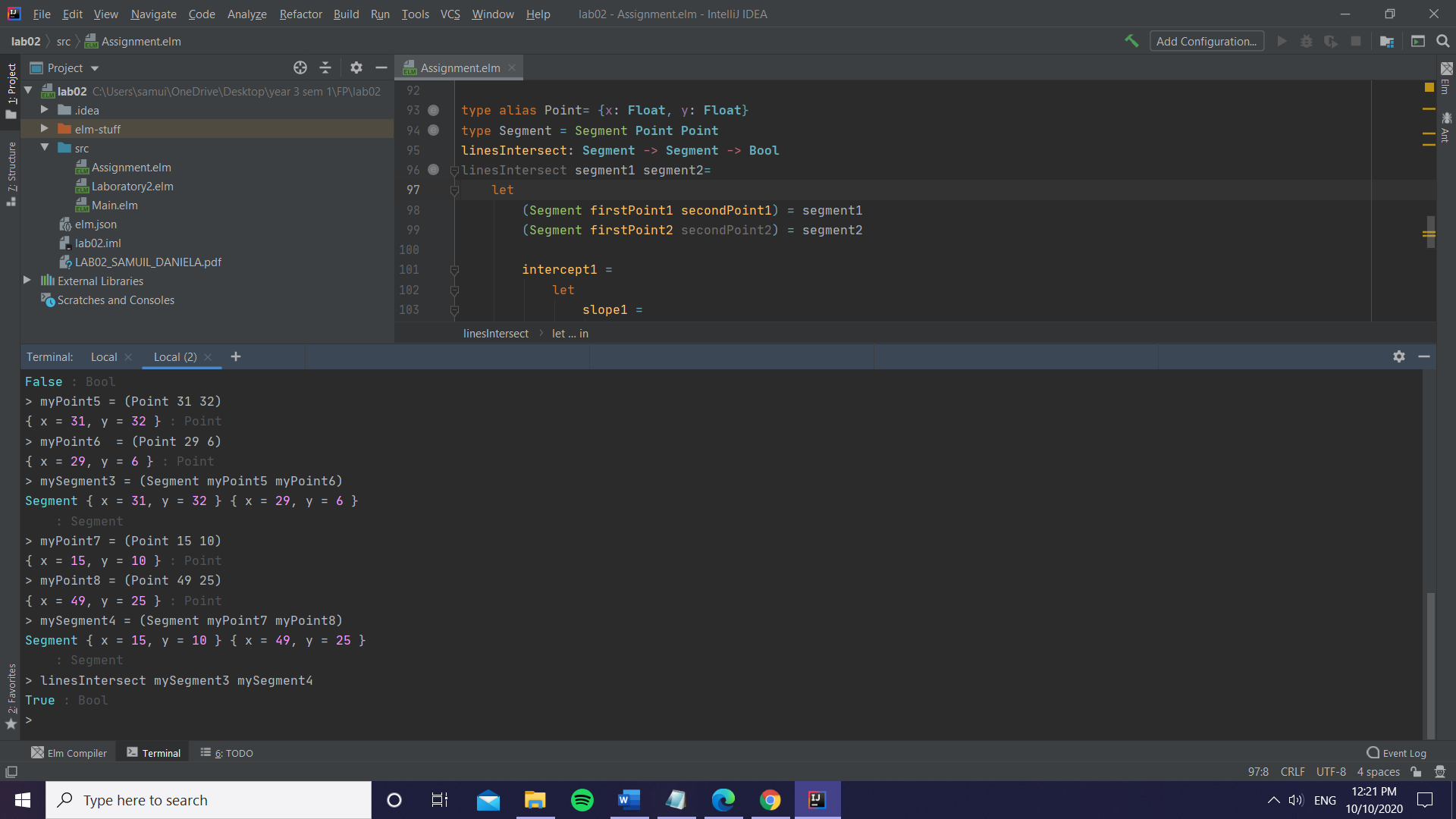


type alias Point= {x: Float, y: Float}

type Segment = Segment Point Point

linesIntersect: **Segment** -> **Segment** -> **Bool**linesIntersect segment1 segment2=  
 let  
 (Segment firstPoint1 secondPoint1) = segment1  
 (Segment firstPoint2 secondPoint2) = segment2  
  
 intercept1 =  
 let  
 slope1 =  
 let  
 (Segment point1 point2) = segment1  
 in  
 (point2.y - point1.y) / (point2.x - point1.x)  
  
 in  
 firstPoint1.y - slope1\*firstPoint1.x  
  
 intercept2 =  
 let  
 slope2 =  
 let  
 (Segment point1 point2) = segment2  
 in  
 (point2.y - point1.y) / (point2.x - point1.x)  
  
 in  
 firstPoint2.y - slope2\*firstPoint2.x  
  
 slope\_1 =  
 let  
 (Segment point1 point2) = segment1  
 in  
 (point2.y - point1.y) / (point2.x - point1.x)  
 slope\_2 =  
 let  
 (Segment point1 point2) = segment2  
 in  
 (point2.y - point1.y) / (point2.x - point1.x)  
  
 x\_intersectionPoint = (intercept2 - intercept1) / (slope\_1 - slope\_2)  
 y\_intersectionPoint = (slope\_1\*x\_intersectionPoint) + intercept1  
  
 in  
 if slope\_1==slope\_2 then False else  
 if (firstPoint1.x > secondPoint1.x) then  
 if (x\_intersectionPoint>secondPoint1.x && x\_intersectionPoint<firstPoint1.x) then  
 if(firstPoint1.y > secondPoint1.y) then  
 if (y\_intersectionPoint>secondPoint1.y && y\_intersectionPoint<firstPoint1.y) then True else False  
 else  
 if (y\_intersectionPoint<secondPoint1.y && y\_intersectionPoint>firstPoint1.y) then True else False  
 else False  
 else  
 if (x\_intersectionPoint<secondPoint1.x && x\_intersectionPoint>firstPoint1.x) then True else False





trailingZeros: **Int** -> **Int**trailingZeros nr =  
 let  
 number = nr  
 in  
 if number == 0 then 0 else trailingZeros (number // 5) + number // 5

