Q1
Coding

```
mystery :: Integer -> Integer -> Integer -> Bool
mystery m n p = not ((m==n) && (n==p))

main = do
        print(mystery 103 103 102)
```

This is taking 3 integer input of m, n, p and return in boolean whether it's true or false.
Integer -> Integer -> Integer
These three integers are number of input

 ->Bool
This is output which is returning boolean

This function says if m is not equal to n and if n is not equal to p. It returns true otherwise it will return false.

If i do

```
Main = do
        print(mystery 103 103 102)
```

 this returns true since m and n equal however n and p is not equal so it return true.

It returns false when all those three numbers are equal like

```
Print(mystery 103 103 103)
```

This returns false

Q2
Coding

```
mintwo :: Int -> Int -> Int
mintwo x y = if x < y then x else y

minThree :: Int -> Int -> Int -> Int
minThree x y z = mintwo (mintwo x y) z

main = do
   print (minThree 103 103 102)
```

This is taking 2 integer input and returns integer value. This takes two integer value and check which one is smaller value and should print out the smaller value.

```haskell
main = do
   print (mintwo 103 102)
```

If I do this, it returns the integer value of 102 since 103 > 102.

```haskell
minThree :: Int -> Int -> Int -> Int
minThree x y z = mintwo (mintwo x y) z
```

Once we found mintwo, we can use mintwo function to find minThree

(mintwo x y)
Find the smaller value between x and y and do
mintwo (mintwo x y) z
To get smaller value between z and (mintwo x y)

```haskell
main = do
   print (minThree 103 103 102)
```

This prints the smallest value out of those three value which in this case will be 102.

Q3
Coding

```haskell
charToNum :: Char -> Int
charToNum c
   | c >= '0' && c <= '9' = read [c]
   | otherwise = 0

main = do
   print (charToNum '8')
```

This functions takes char as input and gives output as integer.
I need to change char value to integer.
Since this is char which it is 1 bit for each character so it will only have 0 to 9.
Other than 0 to 9, it should print out 0.

c >= '0' && c <= '9'
If c is in between 0 to 9 it’s true otherwise false
If that is true do read[c] to get that value as output. We initially set the output value as integer so it should return integer.

I put ‘8’ as char and return 8 as integer output.

Q4
Coding

```haskell
romanDigit :: Char -> String
romanDigit n
    | n == '1' = "I"
    | n == '2' = "II"
    | n == '3' = "III"
    | n == '4' = "IV"
    | n == '5' = "V"
    | n == '6' = "VI"
    | n == '7' = "VII"
    | n == '8' = "VIII"
    | n == '9' = "IX"
    | otherwise = "only 1 digit number"

main = do
  print(romanDigit '8')
```

This takes input as char and output as string.
Since this is char to string, it can only have one digit as input.

If n is equal to specific one digit number as char then it returns string value in romanDigit.

If I print that it returns VIII as output

Q5
Coding

```
mintwo :: Float -> Float -> Float
mintwo x y = if x < y then x else y

maxtwo :: Float -> Float -> Float
maxtwo x y = if x > y then x else y

smallerRoot :: Float -> Float -> Float -> Float
smallerRoot a b c = if b^2 - 4*a*c < 0 then 0 else mintwo ((-b - sqrt(b^2 - 4*a*c)) / (2*a))
((-b + sqrt(b^2 - 4*a*c)) / (2*a))

largerRoot :: Float -> Float -> Float -> Float
largerRoot a b c = if b^2 - 4*a*c < 0 then 0 else maxtwo ((-b - sqrt(b^2 - 4*a*c)) / (2*a))
((-b + sqrt(b^2 - 4*a*c)) / (2*a))

main = do
  print(smallerRoot 1 13 12)
  print(largerRoot 1 13 12)




smallerRoot, largerRoot :: Float -> Float -> Float -> Float
```

It takes 3 float input values and output the 1 float value.

```
mintwo :: Float -> Float -> Float
mintwo x y = if x < y then x else y

maxtwo :: Float -> Float -> Float
maxtwo x y = if x > y then x else y
```

I used this code to find the maximum and minimum float value from 2 float input value.

```
smallerRoot :: Float -> Float -> Float -> Float
smallerRoot a b c = if b^2 - 4*a*c < 0 then 0 else mintwo ((-b - sqrt(b^2 - 4*a*c)) / (2*a))
((-b + sqrt(b^2 - 4*a*c)) / (2*a))
```

b^2 - 4*a*c has to be bigger than 0 since we use root that negative number can't be inside the root. So if it's lower than 0 then it returns 0 otherwise we do the quadratic function to calculate the min value. I used mintwo since we only need to compare from two values. When calculation of square root is positive and one with negative. Find the min value from those two values. I used it same for maxtwo, get the max value from one with positive and one with negative.

```
print(smallerRoot 1 13 12)
print(largerRoot 1 13 12)
```

It returns -12 and -1 respectively.