Q.N 3.1a)

Def^n

$X_1 \times \ldots \times X_n = \{(x_1, \ldots x_n) \mid \forall i \; 1 \le i \le n \Rightarrow x_i \in X_i\}$

To prove:

a) $(A \times B) \cap (C \times D) = (A \times C) \cap (B \times D)$

Taking L.H.S

$\exists x, y \in (A \times B) \cap (C \times D)$

$\Leftrightarrow (x, y) \in (A \times B)$ and $(x, y) \in (C \times D)$

$\Leftrightarrow (x \in A, y \in B)$ and $(x \in C, y \in D)$ [$\because$ from def^n]

$\Leftrightarrow x \in A$ and $y \in B$ and $x \in C$ and $y \in D$

$\Leftrightarrow x \in A$ and $x \in C$ and $y \in B$ and $y \in D$ [rearrangement]

$\Leftrightarrow x \in (A \cap C)$ and $y \in (B \cap D)$

$\Leftrightarrow x \in (A \cap C), y \in (B \cap D)$

$\Leftrightarrow (x, y) \in (A \cap C) \times (B \cap D)$

Hence,

$(A \times B) \cap (C \times D) = (A \times C) \cap (B \times D)$

QN 3.1 b)

We can disprove $(A \cup B) \times (C \cup D) = (A \times C) \cup (B \times D)$ by a counter example,

let's take

$A = \{1, 2\}$, $B = \{3, 4\}$

$C = \{4, 6\}$, $D = \{2, 8\}$

Computing L.H.S

$(A \cup B) \times (C \cup D)$

$= \{1, 2, 3, 4\} \times \{2, 4, 6, 8\}$

$= \{(1,2), (1,4), (1,6), (1,8)$
$(2,2), (2,4), (2,6), (2,8)$
$(3,2), (3,4), (3,6), (3,8)$
$(4,2), (4,4), (4,6), (4,8)\}$

Now,

Computing R.H.S

$(A \times C) \cup (B \times D)$

$= (\{1, 2\} \times \{4, 6\}) \cup (\{3, 4\} \times \{2, 8\})$

$= \{(1,4), (1,6), (2,4), (2,6)\} \cup \{(3,2), (3,8), (4,2), (4,8)\}$

$= \{(1,4), (1,6), (2,4), (2,6)\ (3,2), (3,8), (4,2), (4,8)\}$

As L.H.S $\neq$ R.H.S

We can easily say that $(A \cup B) \times (C \cup D) \neq (A \times C) \cup (B \times D)$ in general

Q: N 3.2a)

$R = \{(a,b) \mid a, b \in \mathbb{Z} \wedge |a-b| \leq 3\}$

Checking if R is reflexive,

for being reflexive

$\forall a \in \mathbb{Z}$ s.t. $(a,a) \in R$

If $(a,a) \in R$ then it must satisfy,

$|a-a| \leq 3$

$\Rightarrow |0| \leq 3$

$\Rightarrow 0 \leq 3$ (True)

So, we can conclude that R is reflexive

Now, checking if R is symmetric,

for being

$\forall a, b \in \mathbb{Z}.\ (a,b) \in R \Rightarrow (b,a) \in R$

Then $|b-a| \leq 3$ must be true,

Taking, $|a-b| \leq 3$

$\Rightarrow |b-a| \leq 3$ ($\because |a-b| = |b-a|$)

(True)

So, we can conclude that R is symmetric.

Lastly, checking if R is transitive,

for being transitive

$\forall a, b, c \in \mathbb{Z} . ((a,b) \in R \land (b,c) \in R) \Rightarrow (a,c) \in R$

For the relation R to be transitive the following must hold,

$(|a-b| \leq 3) \land (|b-c| \leq 3) \Rightarrow (|a-c| \leq 3) \; \forall a,b,c \in \mathbb{Z}$

We can disprove the above statement by following example,

lets take $a = 10$, $b = 7$ and $c = 5$

$|a-b| \leq 3 \Rightarrow (10-7) \leq 3 \Rightarrow 3 \leq 3$ (True)

and,

$|b-c| \leq 3 \Rightarrow (7-5) \leq 3 \Rightarrow 2 \leq 3$ (True)

Now,

$|a-c| \leq 3 \Rightarrow (10-5) \leq 3 \Rightarrow 5 \leq 3$ (False)

Hence,

By giving a counter example we can conclude that R is non-transitive.

Q.N 3.2 b)

$R = \{(a,b) \mid a,b \in Z \wedge (a \bmod 10) = (b \bmod 10)\}$

Checking if R is reflexive,

for being reflexive

$\forall a \in Z : (a,a) \in R$

If $(a,a) \in R$ then it must satisfy,

$(a \bmod 10) = (a \bmod 10)$ (True)

So, it is reflexive

Checking if R is symmetric,

for being symmetric,

$\forall a,b \in Z . (a,b) \in R \Rightarrow (b,a) \in R$

If $(a,b) \in R$ then it must hold that,

$(a \bmod 10) = (b \bmod 10) \Rightarrow (b \bmod 10) = (a \bmod 10)$

As it is equality the above statement is true.

So, it is symmetric.

Checking if R is ~~transitive~~ transitive,

for being transitive,

$\forall a,b,c \in \mathbb{Z}. ((a \bmod 10) = (b \bmod 10)) \wedge ((b \bmod 10) = (c \bmod 10))$
$\Rightarrow ((a \bmod 10) = (c \bmod 10))$

Taking $(a \bmod 10) = (c \bmod 10)$
$\Rightarrow (b \bmod 10) = (c \bmod 10)$ (As $(a \bmod 10) = (b \bmod 10)$)
(True)

Hence, we can say that R is transitive.

QN 3.3) Solution:

We have,

```
cnt :: Eq a => a -> [a] -> Int
cnt x [] = 0
cnt x (y:ys)
    | x == y = 1 + (cnt x ys)
    | otherwise = cnt x ys
con :: [a] -> [a] -> [a]
con [] ys = ys
con (x:xs) ys = x : (con xs ys)
```

The cnt function is for counting the number of appearance of value of x in a list. And, the con function is for joining two lists into one single list containing the values of those two list.

To Prove:

cnt x (con s t) == (cnt x s) + (cnt x t)

Using the mathematical induction,

Basic Steps: Let's consider the empty list.

Taking s = []

So our expression would be,

=> cnt x (con [] t) == (cnt x []) + (cnt x t)

=> cnt x t == 0 + (cnt x t)

=> cnt x t == cnt x t   (True)

[As cnt returns Int, so numeric operations validate]

Inductive Steps:

Assumption: cnt x (con s t) == (cnt x s) + (cnt x t)

Let’s take a list k such that the length(k) = (length(s)+1), and the extra value that k contains be c.

```
To prove:

cnt x (con k t) == (cnt x k) + (cnt x t)
So, our expression would be,
=> cnt x (con k t) == (cnt x k) + (cnt x t)
=> cnt x (k ++ t) == (cnt x k) + (cnt x t)  [As, (con k t) = k ++ t]
```

```
From our program,
    If x == c, the above expression is equal to
    => 1 + cnt x (s ++ t) == (1 + (cnt x s)) + (cnt x t)
    => 1 + cnt x (con s t) == 1 + (cnt x s) + (cnt x t)
                                [As, s ++ t = (con s t)]
    => 1 + (cnt x s) + (cnt x t) == 1 + (cnt x s) + (cnt x t)  (True)
                                [From our assumption,
                                cnt x (con s t) == (cnt x s) + (cnt x t)]

    Otherwise,
    => 0 + cnt x (s ++ t) == (0 + (cnt x s)) + (cnt x t)
    => 0 + cnt x (con s t) == 0 + (cnt x s) + (cnt x t)
                                [As, s ++ t = (con s t)]
    => 0 + (cnt x s) + (cnt x t) == 0 + (cnt x s) + (cnt x t)  (True)
                                [From our assumption,
                                cnt x (con s t) == (cnt x s) + (cnt x t)]

Hence, we can conclude inductively that,
            cnt x (con s t) == (cnt x s) + (cnt x t)
```

QN 3.4a) solution:

```
rotate :: Int -> [a] -> [a]

rotate _ [] = []

rotate x xs

       | (x `mod` length(xs)) == 0 = xs

rotate x ys = rotate ((x-1) `mod` length(ys)) (tail ys ++ [head ys])
```

QN 3.4b) solution:

```
--please note that you must load the rotate function from question 3.4a in order
to run this program successfully.

circle :: [a] -> [[a]]

circle [] = []

circle xs = [(rotate x xs) | x<-[0..(length(xs)-1)]]
```