Midterm #2 Cheatsheet

Notes:

First :
- → no base case
- → solve left downest possible
- → move right until meeting left's conclusion
- → write the reverse steps

Second :
- → no inductive step, write Inductive Hypothesis
- → no left side, may want/need to apply IH to get to right side
- → no right side, continue until matching left side
- → write the reverse steps

Third :
- → if cannot match 100% of the left and right side, do case analysis on concrete/common variables

example: case (h = x) is true
case (h = x) is false } see 1c. example

Other: If cannot find way from left to move forward, then start from the right side. can also go from right to left

2c) Prove that the identity function on integer given by:
let id (x:int) : int = x constitutes a valid instance for involution. Write the instance and prove any required properties.

"Discord": create new instance/module that satisfies the properties of involution, such as defining both t and inv

```
module type Involution = sig
  type t
  (** Requires that inv is its own inverse:
  [inv (inv x) = x] **)
  val inv : t -> t
end
```

```
module Identity = struct
    type t = int
    let inv = id
end
```

inv (inv x) = x

id x = x (given)

Identity.inv (Identity.inv x)
= {inv def twice}
id (id x)
= {id def}
id (x)
= {id def}
x

Use given/proven proofs!

own lemma : or x x = x   case x = true
case x = false

If statements : {conditional apply}

or x y = or y x
Cases:
case: x = true y = true
case: x = true y = false
case: x = false y = true
case x = false y = false

Induction on lst =	case lst = [] case lst = h::tl , write IH
Induction on b =	case b = zero case b = S d , write IH

proof by cases :	case x = true case x = false

Coming up with cases :
- look at function match statements and struct/type

```
type expr
  = Int of int
  | Plus of t * t
let rec mirror t =
    match t with
    | Int n -> Int n        <- base case -> case : x = Int n
    | Plus (l,r) -> Plus (mirror r, mirror l)  <- Inductive case case: x = Plus (l,r)
```

Inductive Hypothesis:
- For lst: replace original proof's lst with tl
- Can have more than 1 IH:

example: mirror (mirror x) = x   case x = Plus (a,b)
IH1 : mirror (mirror a) = a
IH2 : mirror (mirror b) = b

1c. Write the inductive case on this page. Be sure to clearly write your inductive hypothesis:

Unsure if doing IH correctly? what's different from to-be-proved equation vs. IH???

Prove that:
or (contains x lst1) (contains x lst2) = (contains x (append lst1 lst2))
use induction on lst1.

Inductive step: lst1 = h::tl

Inductive hypothesis: or (contains x' lst1')(contains x' lst2') = (contains x' (append lst1' lst2'))

case: lst1 = h::tl

left:

or (contains x lst1)(contains x lst2)

= {cases}

or (contains x h::tl)(contains x lst2)

= {cts-c}

or (if hd = x then true else contains x tl)(contains x lst2)

"new left side"

= {???}

= {reverse match from right side}

if hd = x then true else contains x ~~tl~~ (append tl lst2)

= {cts-c}

(contains x (hd::append tl lst2))

= {ap-c}

(contains x (append h::tl lst2))

= {case}

(contains x (append lst1 lst2))

forgot to apply IH

No need to apply IH if have {???} for case analysis distinction

right:

(contains x (append lst1 lst2))

= {case}

(contains x (append h::tl lst2))

= {ap-c}

(contains x (hd::append tl lst2))

contains x (hd::tl)

= {cts-c}

(if hd = x then true else contains x ~~tl~~ (append tl lst2))

"new right side"

Slides

IH: or (contains x tl)(contains x lst2)
= contains x (append tl lst2)

replaced lst1 with tl for the inductive hypothesis???

case: hd = x is true

or (if hd = x then true else contains x tl)(contains x lst2)

= {case}

or (if true then true else contains x tl)(contains x lst2)

= {ite-t}

or true (contains x lst2)

= {or-t}

true

= {reverse from right}

= {ite-t}

if true then true else contains x (append tl lst2))

= {case}

if h = x then true else contains x (append tl lst2))

right:

if hd = x then true else contains x (append tl lst2))

= {case}

if true then true else contains x (append tl lst2))

case: hd = x is false

or (if hd = x then true else contains x tl)(contains x lst2)

= {case}

or (if false then true else contains x tl)(contains x lst2))

= {ite-f}

or (contains x tl)(contains x lst2))

= {inductive hypothesis}

(contains x (append tl lst2))

= {reverse from right}

= {ite-f}

if false then true else contains x (append tl lst2))

= {case}

if hd = x then true else contains x (append tl lst2))

right:

if hd = x then true else contains x (append tl lst2))

= {case}

if false then true else contains x (append tl lst2))