1 - Type Inference (20 points)

1) Suppose we have the following expressions (we omit some information and we replace it with #n, where n is some positive integer).

```
a : #1 list
b : #2
let c = b :: a List
let d = (1, true) :: c
```

where the last expression type checks without error.

- What is the type #1? int * bool
- What is the type #2? Tht * bool
- What is the type of the expression ([d])?

2) Suppose we have the following expressions (we omit some information and we replace it with #n, where n is some positive integer):

```
a : int list
b : #1
c : #2 (int*int) Tht int List
let (x,y) = b in (x :: a, [x + y] @ c)
```

where the last expression type checks without error.

- What is the type #1? (int * int).
- What is the type #2? Int list.
- What is the type of the expression ([[x]], c:: [])?
 Int List List.
 Int List List x int List List.

(3) Suppose x_0 could be matched as below without error

• What is the type of the variable x_0 ?

bool yst list

• Now consider the first match for x_0 , where x_0 is getting pattern matched with []. Let us suppose that we replace [true] with ["true"]. Do you think this replacement will result in some kind of error indicating that the types are no longer consistent? Answer this question either with a YES or a NO and explain why.

No. adding " will change its type from bool to String, but it will still be consistent, because X. will change to String list list and I true"] will become string list.

4) Consider the following program:

• What is the type of a? Briefly explain your reasoning.

• What is the type signature of foo? Briefly explain your reasoning.

Because a is int, Is is int list.

Therefore, had is int, this int list. -> halt a = int + int = int

aux will keep tec until the int a,

So for is int list -> int.

2 - Pattern matching (18 points)

1) Consider the following expressions

```
a: bool true/fouse
b: bool

C: bool

FIT FIF FFF FFT

(8) typus
```

Complete the following match so that every possible expression that replaces a, b and c in the matching statement is matched. You must ensure that you have exhausted all possible cases for a, b and c. Please do not use underscore in your match.

2) Consider the following expressions

Complete the following match so that every possible expression that replaces a and b in the matching statement is matched, distinguishing cases for lists and tuples — no need to distinguish cases for int and float. Please do not use underscore in your match.

```
match (a,b) with

([[],()),[]-)

(x::\text{$\(x\)},()),[]-)

([[],()),\text{$\(y\)}:\text{$\(y\)},-)

(\text{$\(x\)}:\text{$\(y\)},\text{$\(y\)},\text{$\(y\)}:\text{$\(y\)},-)
```

3) Suppose that 1 is an expression of type (t list) list, and consider the following pattern matching.

Is this match exhaustive? That is, does this match explore all the possible forms of 1?

☐ No

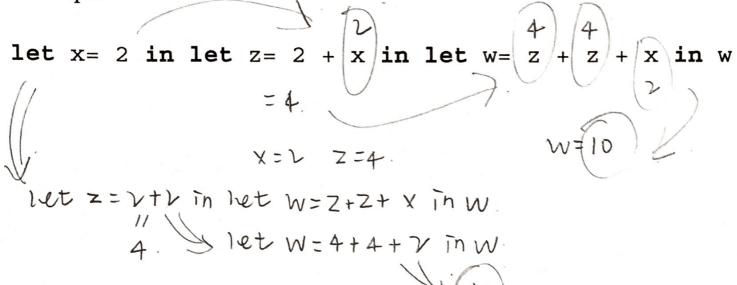
4) Suppose 1 is an expression of type (int,int) list, and consider the following pattern matching.

Is this match exhaustive? That is, does this match explore all the possible forms of 1?



3 - Let-binding reduction (12 points)

1) Reduce the following expression to a value. Make sure that you show all the steps:



2) Reduce the following expression to a value. Make sure that you show all the steps:

The steps.

$$x = 10$$

let $x = 7+3$ in let $(z,y) = (x + x, 3)$ in let

 $(x,u,w) = (5/y,z+z)$ in $u + x$
 $y = 3$
 $y = 40$
 $y = 3$
 $y = 3$
 $y = 40$
 $y = 3$
 $y = 3$
 $y = 40$
 $y = 3$
 $y = 40$
 $y = 3$
 $y = 3$
 $y = 40$
 $y = 3$
 $y = 40$