Practice Mid-Term

Add a try-catch block to the grammar in IA-2

Show some example code that uses the try-catch block.

Show the parse tree for the example code.

Show the canonical derivation of the example code.

```
1  x : integer := 1
2  y : integer := 2
3
4  procedure add
5    x := x + y
6
7  procedure second(P : procedure)
8    x : integer := 2
9    P()
10
11 procedure first
12   y : integer := 3
13   second(add)
14
15 first()
16 write integer(x)
```

What does this program print if the language uses static scoping?

What does it print if the language uses dynamic scoping with deep binding?

What does it print if the language uses dynamic scoping with shallow binding?

The grammar for IA-2 does not allow for complex boolean expressions.

```
1 A > B && C > D || F == G
```

Extend the grammar to allow these new types of boolean expressions.

(You will need to add new operators for AND and OR)

AND and OR can have the same precendence, but you will need to address associativity.

Write the recursive-decent method for parsing the new production.

Show the parse tree for the above snippet.

Does your complex boolean expression implement left or right associativity?

On onyx, find the source code for the n-queens problem at:

onyx: jmazzare/cs354/scheme/n-queens/

For all global and local defined functions:

Is the function recursive?

If so, is the function tail-recursive?

How do you know?

If it is not tail-recursive, make it tail-recursive.

Start with these functions:

```
1  (define (filter-coords coords-list)
2    (if (null? coords-list) '()
3      (if (safe-place? (car coords-list) placed-queens)
4        (cons (car coords-list) (filter-coords (cdr coords-list)))
5        (filter-coords (cdr coords-list)))))
6
7  (define (gen-rec i)
8    (if (>= i (* n n)) '()
9      (cons
10       (cons (quotient i n) (remainder i n))
11       (gen-rec (+ 1 i)))))
12
```

Make sure the code still runs and gives the correct answer.