CSE 3341, Assignment #2
Due: 10 Feb. '20
20 points.

Suppose we wish to add a '**select**' statement to Core as follows: a **select** statement has the structure

```
select b1 -> S1 || b2 -> S2 || ... || bn -> Sn end;
```

where `b1, ..., bn` are boolean expressions (i.e. ⟨*cond*⟩'s), and `S1, ..., Sn` are ⟨*stmt seq*⟩'s; and the number of `b`'s is equal to the number of `S`'s. We execute the `select` as follows: if `b1` evaluates to **true**, execute `S1` and you are done; if `b1` evaluates to **false**, and `b2` to **true**, execute `S2` and you are done; ...; if `b1, ..., b(n-1)` evaluate to **false** and `bn` to **true**, execute `Sn`; if `b1, ..., bn` all evaluate to **false**, simply go to the next statement that follows the **select** (i.e., the **select**, in this case, is a 'no-op'). Note that the "`||`" is a new terminal symbol that is used to separate each ⟨*cond*⟩,⟨*stmt seq*⟩ pair from the next pair.

Now for the problems:

1. (8 points). Add the **select** statement to Core by modifying the BNF grammar appropriately. (Use "pure BNF", i.e., what we have been using in class. If you were to use "extended" BNF, answering the next question will be much harder.)

2. (12 points). Using the notation from the class notes (copies of slides), write down the *Execute-select-statement* procedure and any other procedures you need to add to Core's interpreter to implement the **select** statement. Don't worry about the *Parse-select-statement* procedure.

   You may assume the array representation of the parse tree when answering this question. You may also assume procedures such as *Execute-stmt-seq* or functions like *Eval-Cond* etc. already exist.

Important Note: The problem is *not* asking you to show how you can achieve the *effect* of the *select* statement by using multiple *if* statements or anything like that. You are being asked to introduce a *new* statement into the CORE language so that the CORE programmer can write, as part of his or her CORE program, statements such as:

```
select (X > 0) -> Y = 10; || (Y > 0) -> X = 10; end;
```