

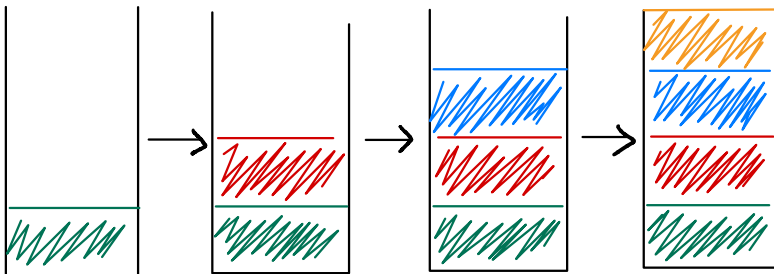


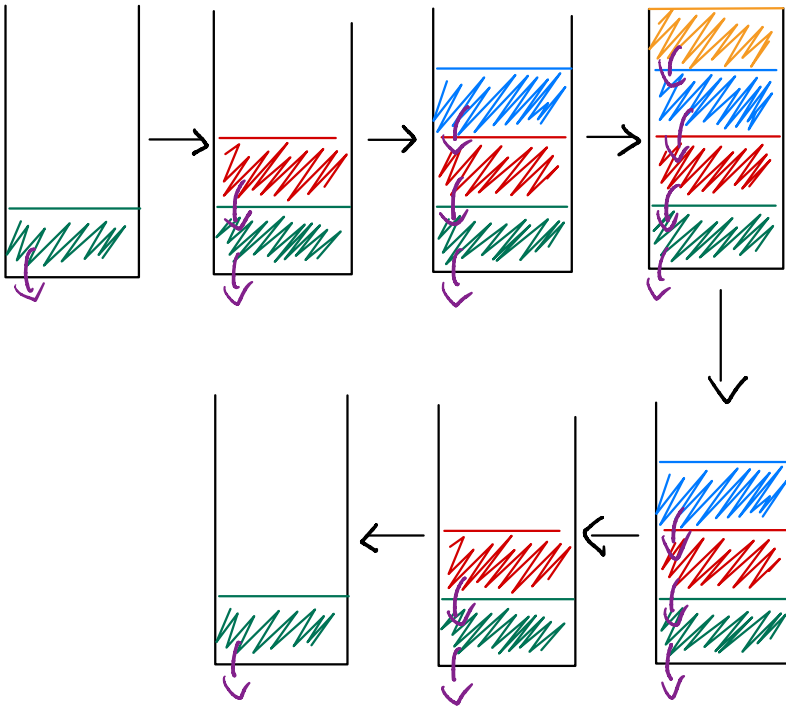
```


[Sun XGreederThcnY[ X:int, y:int) → bool [ {
  let ans: bool = true;
  if (X <= y) [ { ans = true; }
  return ans;
}] ]


```

Note: '[' denote a push and ']' denote a pop.



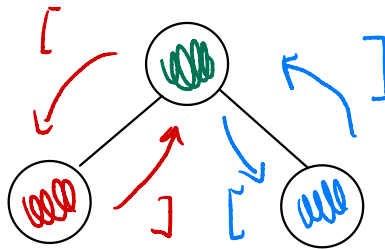


 Green represents the global scope such a scope is often never popped.

 The arrows marked in purple indicate that Symbol Tables hold a reference/pointer to their enclosing scope

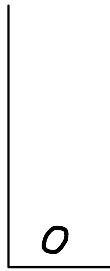
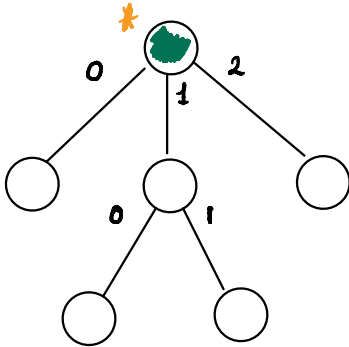
[[{  
 { } - print 10;  
 { }  
 { } - print 20;  
 { } ] ]

Note: This example better demonstrates the Environment tree



Again each node is a Symbol Table (or Environment).

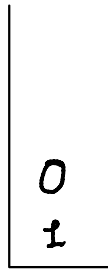
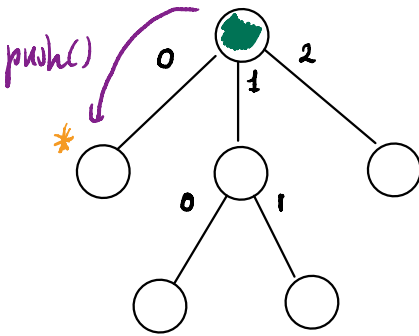
①



Initial State

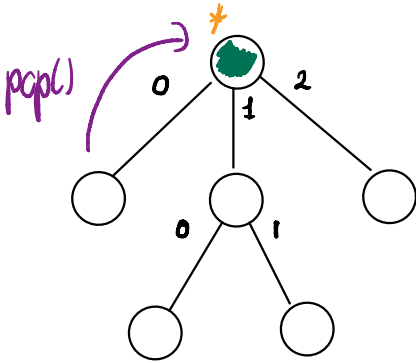
let \* denote the current scope.

②



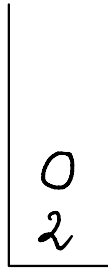
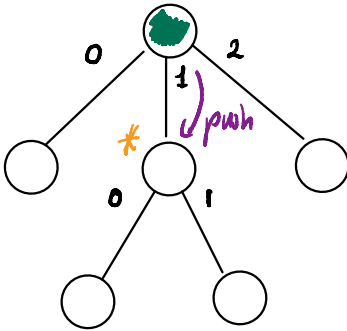
Calling Push():

1. Read the index at the top of the stack into  $n$ .
2. Increment the index at the top of the stack.
3. Push 0 onto the stack.
4. Set the current environment to `current.children[n]`.

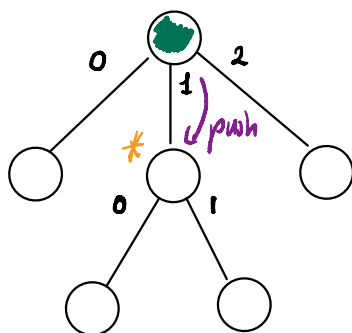


Calling Pop():

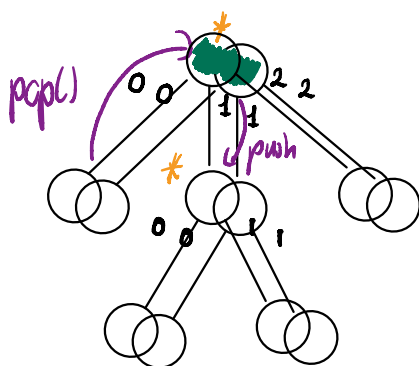
1. Pop from the stack
2. Set the current scope to the enclosing scope



Calling Push(), again:

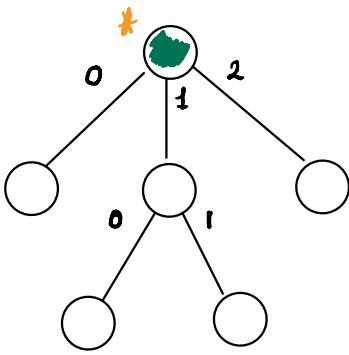


0
2



0
2

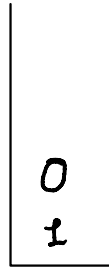
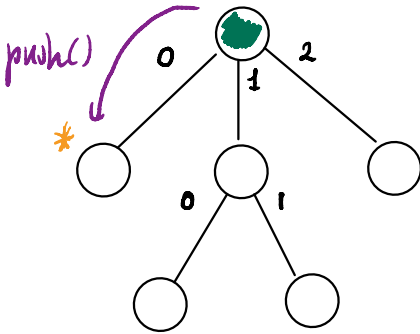
①



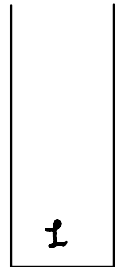
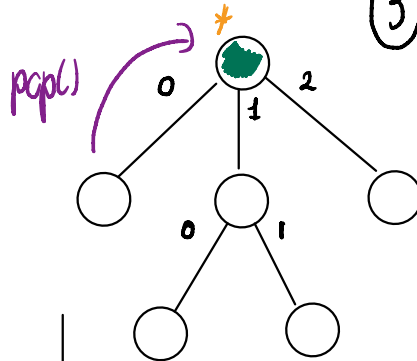
Initial State

let \* denote the current scope.

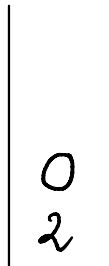
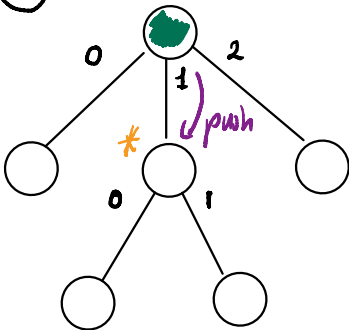
②



③



④



/// The root is the global scope