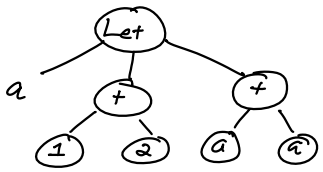
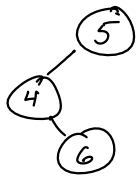
Auklet

Syntax: <expr> ::= 0 | 1 | -1 | 2 | -2 | ...
| after (<expr>)
| before (<expr>)
| <expr> + <expr>
| <expr> - <expr>
| <expr> * <expr>
| let <var> = <expr> in <expr>
| <var>

let a = 1 + 2 in
a + a

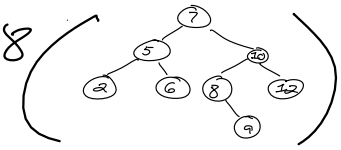


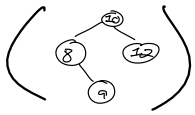
Semantics:
(by example)

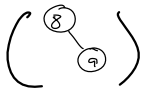


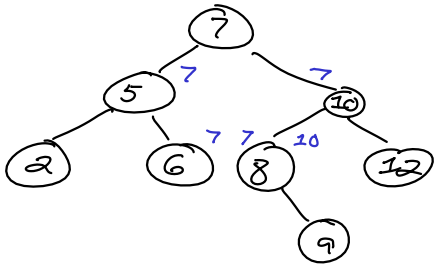
NOT A BST

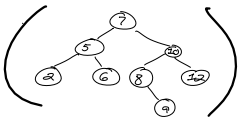
All left descendants have lesser keys; all right descendants have greater keys.

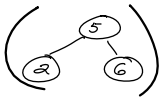
Contains 8 

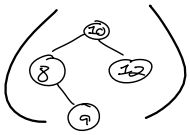
Contains 8 

Contains 8 



check -∞ +∞ 

check -∞ 7  &&

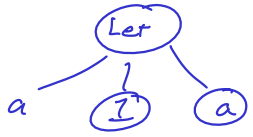
check 7 +∞ 

```
let compile_expression (env:environment)(e:expr) : instruction list =
  match e with
  | EInt n -> [AsmMov(ArgRegister RAX, ArgConstant n)]
  | EVar x ->
      let offs = get_var x env in
      [AsmMov(ArgRegister RAX,
              ArgMemory(MemOffset(RSP, offs)))]
  | ELet(x, e1, e2) ->
      let instrs1 = compile env e1 in
      let (env', offs) = alloc_var x env in
      let instrs2 = [AsmMov(
                      ArgMemory(MemOffset(RSP, offs)),
                      ArgRegister RAX)] in
```

answer will be stored in RAX after instructions run

mov rax, n

let a = 1 in a



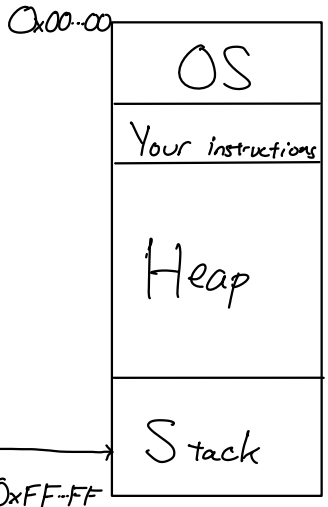
1 → mov rax, 1

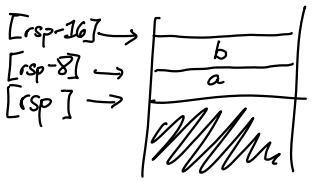
let a → mov [rsp-8], rax

in a → mov rax, [rsp-8]

```
let a = 1 in
let b = 2 in
a
```

Environment $\left(\{ \text{"a"} \mapsto -8 \}, -16 \right)$





NOT TO SCALE