Assignment D

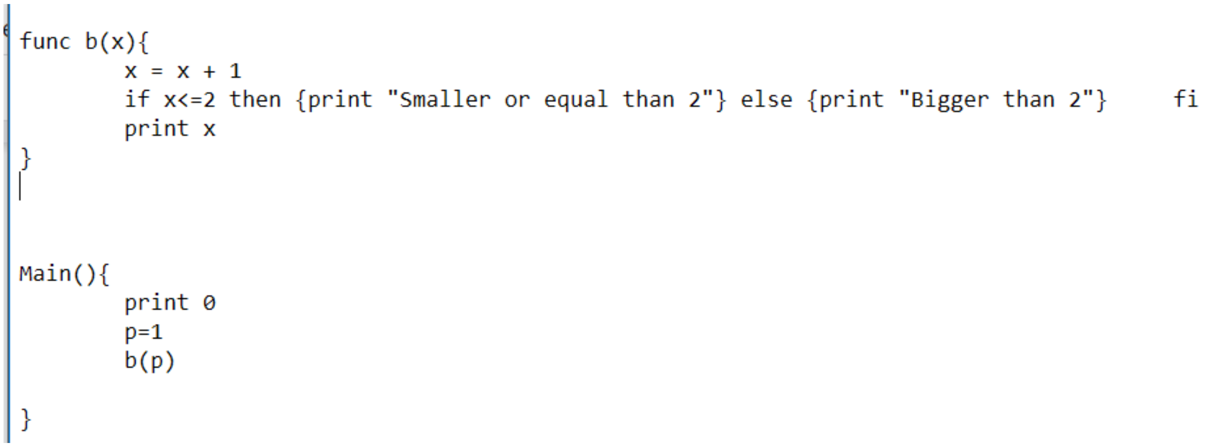
Team: Andreea Matei & Florin Deleanu

Test Cases

Functions

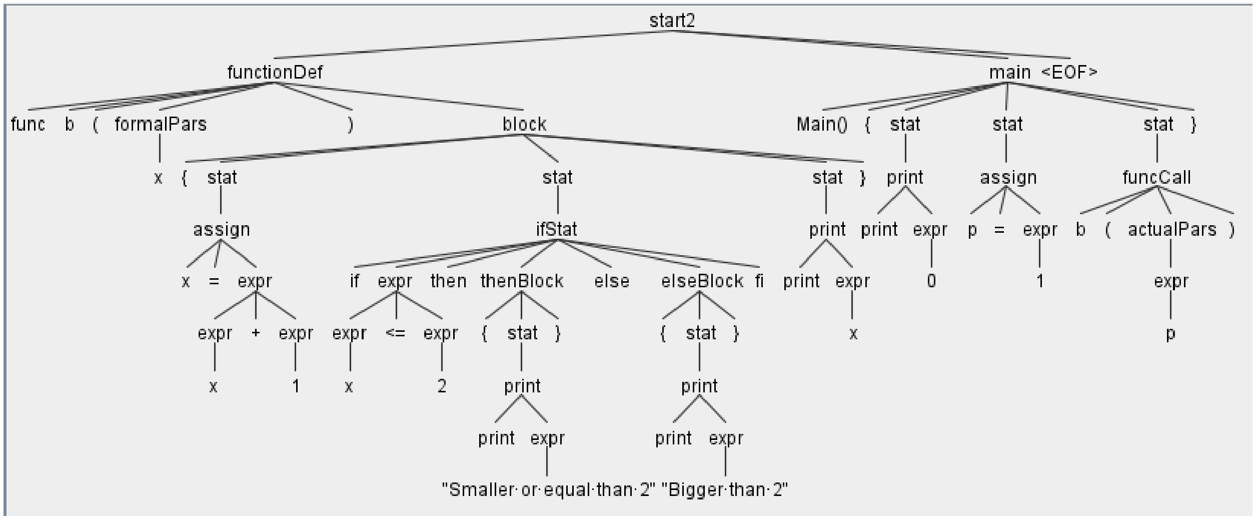
Test 1

Input:

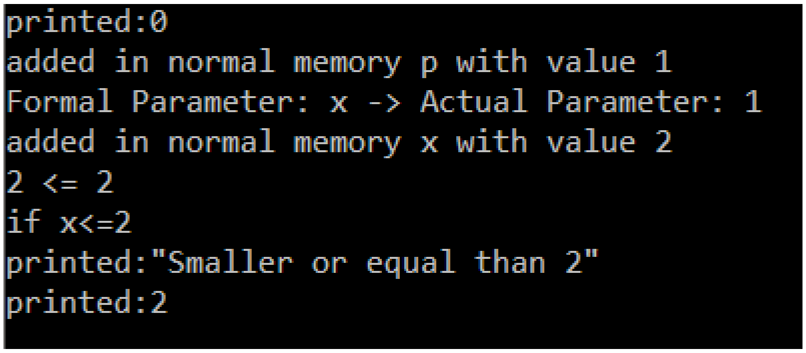


```
func b(x){
        x = x + 1
        if x<=2 then {print "Smaller or equal than 2"} else {print "Bigger than 2"}     fi
        print x
}



Main(){
        print 0
        p=1
        b(p)

}
```

Parse tree:

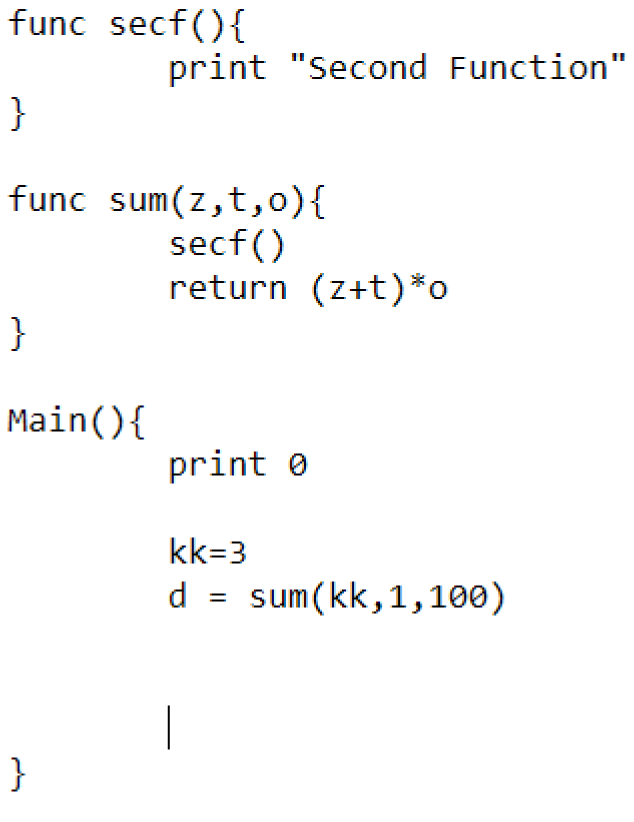


Output:



```
printed:0
added in normal memory p with value 1
Formal Parameter: x -> Actual Parameter: 1
added in normal memory x with value 2
2 <= 2
if x<=2
printed:"Smaller or equal than 2"
printed:2
```

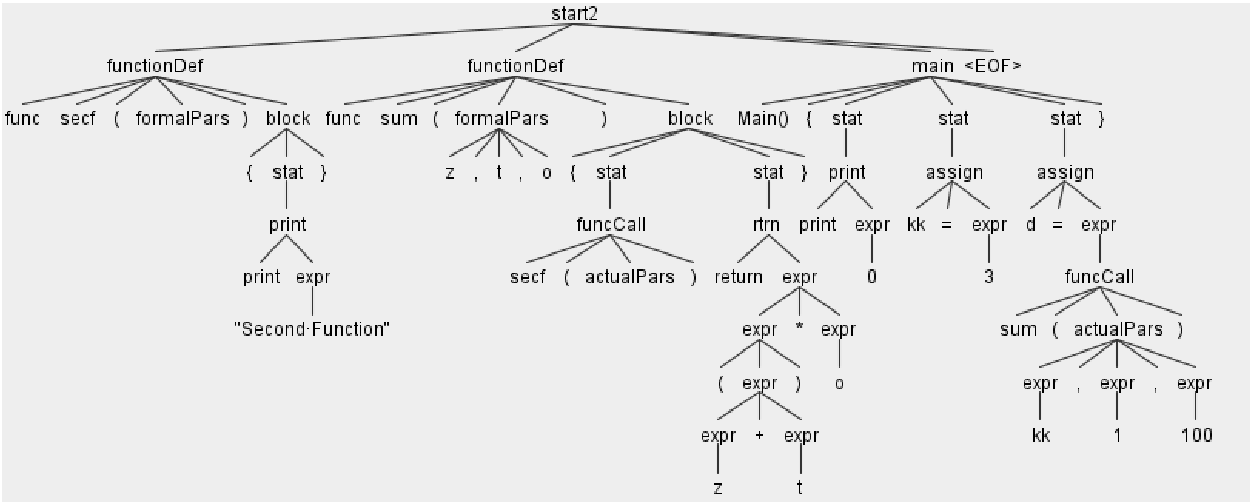
Test 2



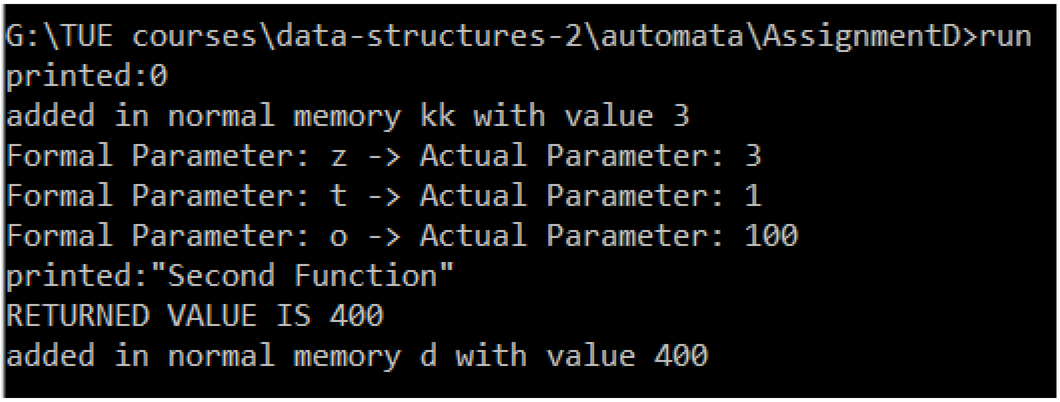
```
func secf(){
        print "Second Function"
}

func sum(z,t,o){
        secf()
        return (z+t)*o
}

Main(){
        print 0

        kk=3
        d = sum(kk,1,100)


        
}
```

Input:

Parse tree:


start2
functionDef functionDef main <EOF>
func secf (formalPars) block func sum (formalPars) block Main() { stat stat stat }
{ stat } z , t , o { stat stat } print assign assign
print funcCall rtrn print expr kk = expr d = expr
print expr secf (actualPars) return expr 0 3 funcCall
"Second·Function" expr * expr sum (actualPars)
(expr) o expr , expr , expr
expr + expr kk 1 100
z t

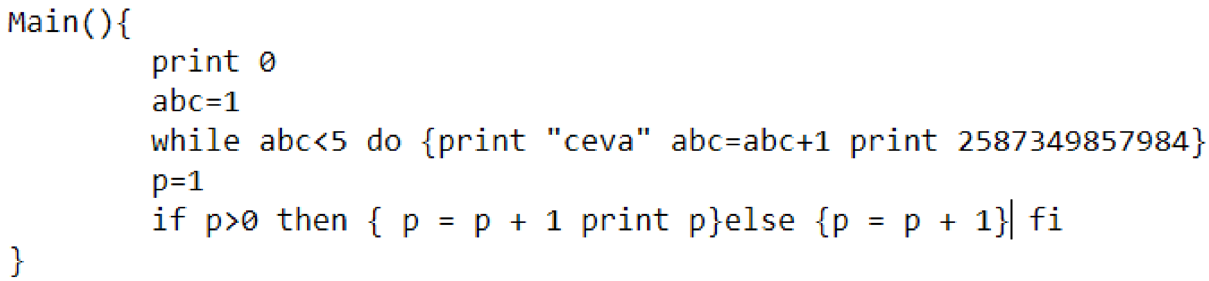
Output:



```
G:\TUE courses\data-structures-2\automata\AssignmentD>run
printed:0
added in normal memory kk with value 3
Formal Parameter: z -> Actual Parameter: 3
Formal Parameter: t -> Actual Parameter: 1
Formal Parameter: o -> Actual Parameter: 100
printed:"Second Function"
RETURNED VALUE IS 400
added in normal memory d with value 400
```

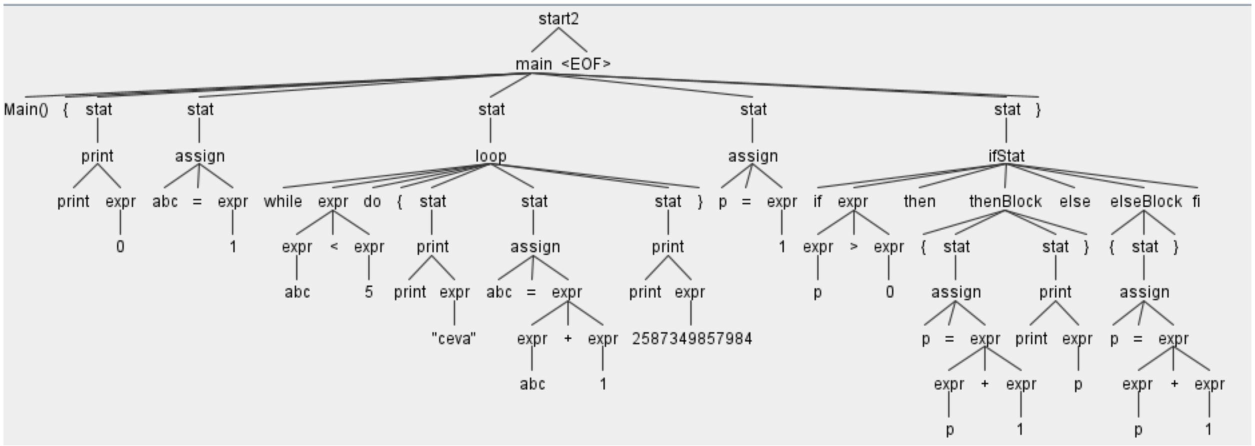
Test 3

Input:



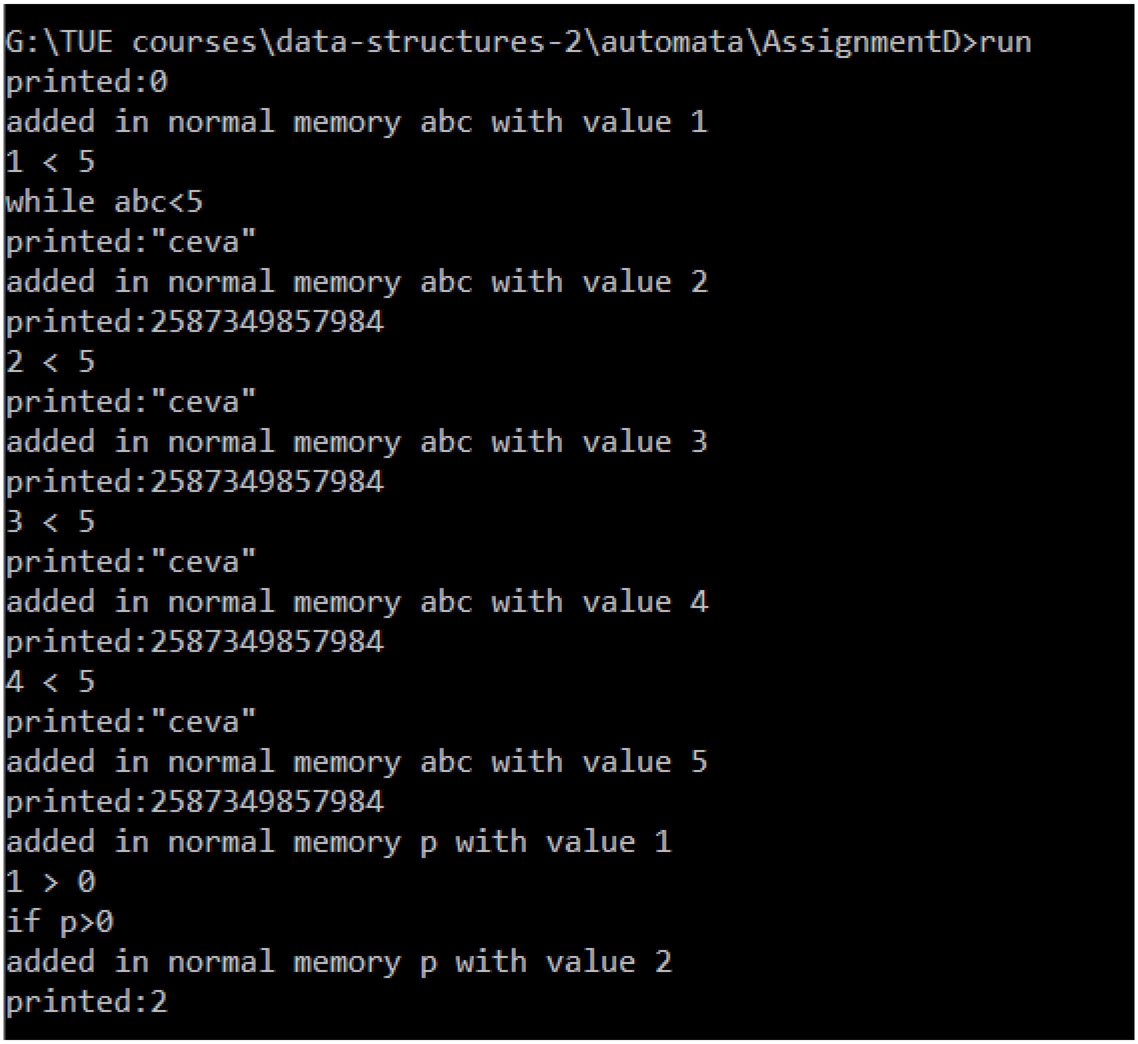
```
Main(){
        print 0
        abc=1
        while abc<5 do {print "ceva" abc=abc+1 print 2587349857984}
        p=1
        if p>0 then { p = p + 1 print p}else {p = p + 1} fi
}
```

Parse tree:



start2
main <EOF>
Main() { stat stat stat stat stat }
print assign loop assign ifStat
print expr abc = expr while expr do { stat stat stat } p = expr if expr then thenBlock else elseBlock fi
0 1 expr < expr print assign print 1 expr > expr { stat stat } { stat }
abc 5 print expr abc = expr print expr p 0 assign print assign
"ceva" expr + expr 2587349857984 p = expr print expr p = expr
abc 1 expr + expr p expr + expr
p 1 p 1

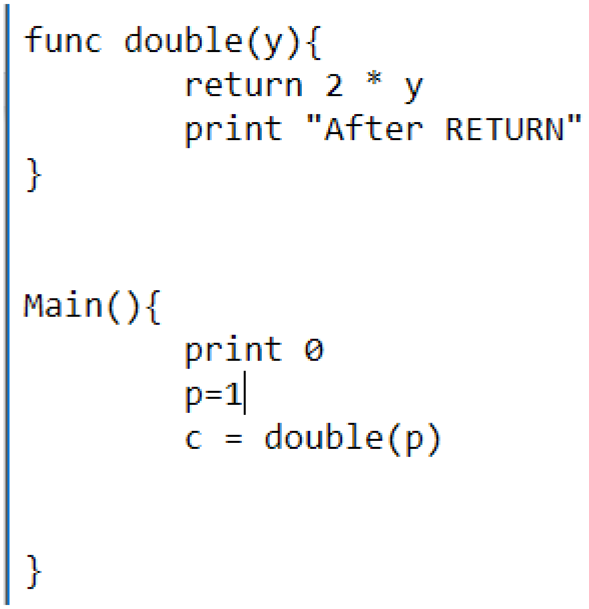
Output:



```
G:\TUE courses\data-structures-2\automata\AssignmentD>run
printed:0
added in normal memory abc with value 1
1 < 5
while abc<5
printed:"ceva"
added in normal memory abc with value 2
printed:2587349857984
2 < 5
printed:"ceva"
added in normal memory abc with value 3
printed:2587349857984
3 < 5
printed:"ceva"
added in normal memory abc with value 4
printed:2587349857984
4 < 5
printed:"ceva"
added in normal memory abc with value 5
printed:2587349857984
added in normal memory p with value 1
1 > 0
if p>0
added in normal memory p with value 2
printed:2
```

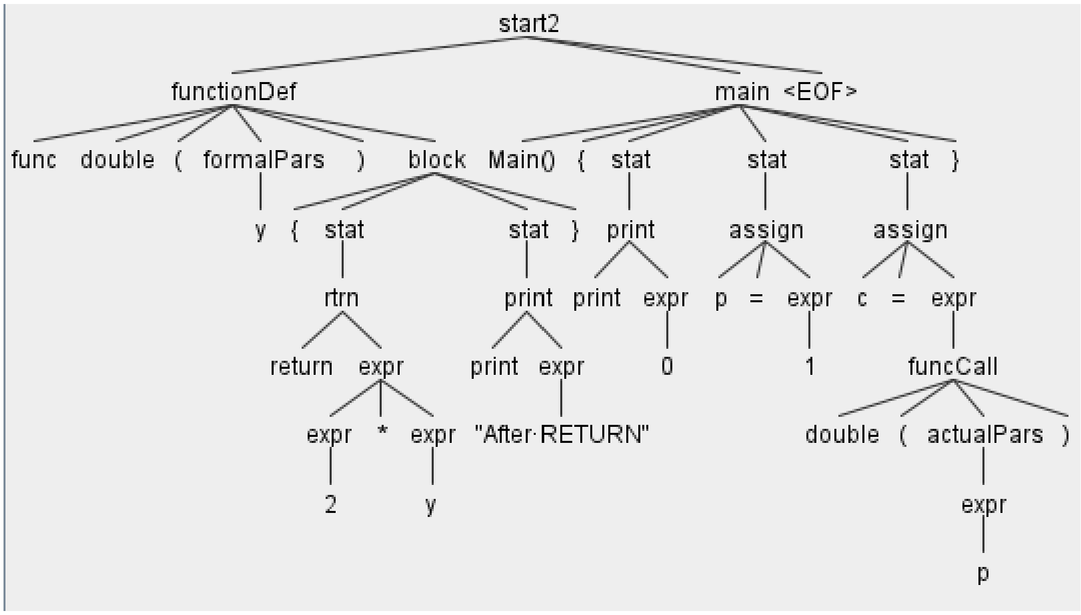
Test 4

Input:

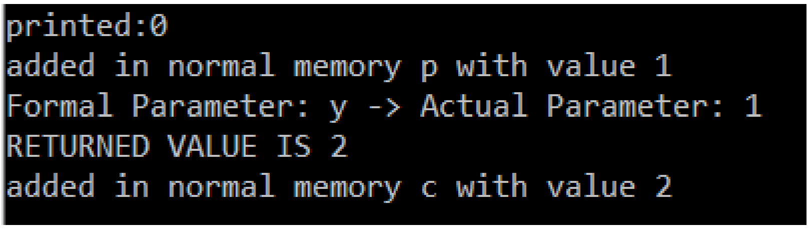


```
func double(y){
        return 2 * y
        print "After RETURN"
}


Main(){
        print 0
        p=1
        c = double(p)



}
```

Parse tree:



Output:

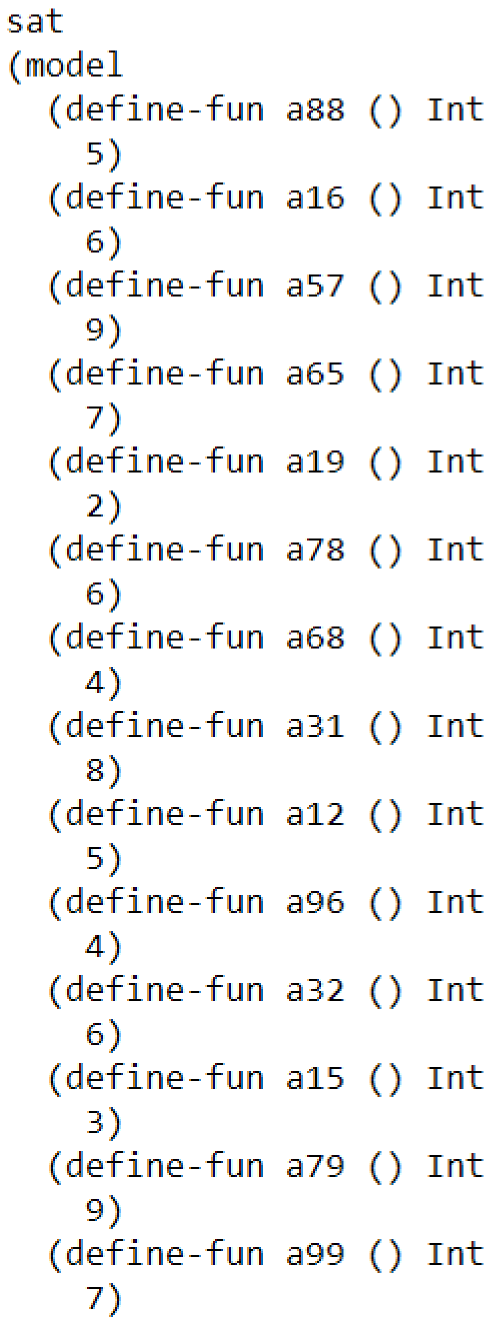


```
printed:0
added in normal memory p with value 1
Formal Parameter: y -> Actual Parameter: 1
RETURNED VALUE IS 2
added in normal memory c with value 2
```

Z3 Output

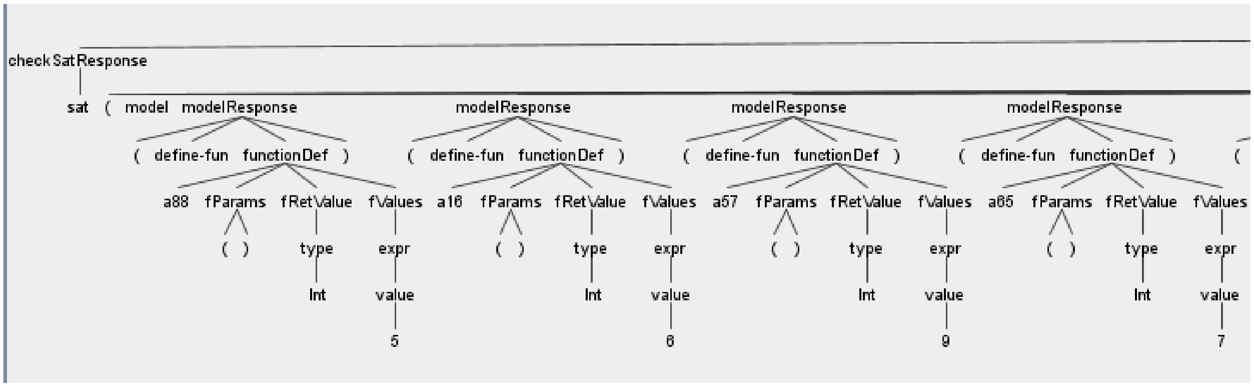
Test 1

Input:

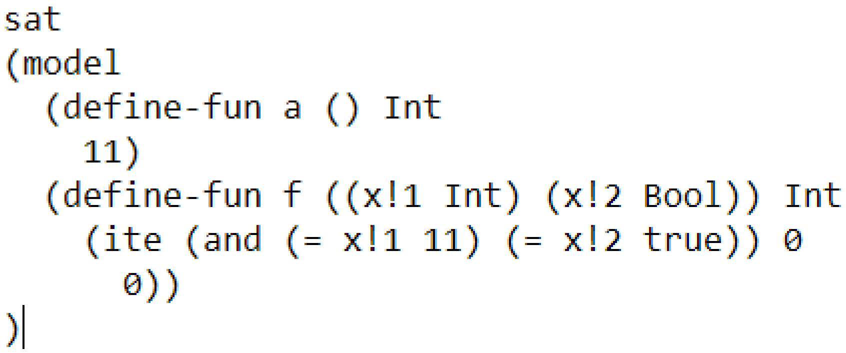


```
sat
(model
  (define-fun a88 () Int
    5)
  (define-fun a16 () Int
    6)
  (define-fun a57 () Int
    9)
  (define-fun a65 () Int
    7)
  (define-fun a19 () Int
    2)
  (define-fun a78 () Int
    6)
  (define-fun a68 () Int
    4)
  (define-fun a31 () Int
    8)
  (define-fun a12 () Int
    5)
  (define-fun a96 () Int
    4)
  (define-fun a32 () Int
    6)
  (define-fun a15 () Int
    3)
  (define-fun a79 () Int
    9)
  (define-fun a99 () Int
    7)
```

Parse tree:


checkSatResponse
sat (model modelResponse
modelResponse
modelResponse
modelResponse
(define-fun functionDef)
(define-fun functionDef)
(define-fun functionDef)
(define-fun functionDef)
(
a88 fParams fRetValue fValues
a16 fParams fRetValue fValues
a57 fParams fRetValue fValues
a65 fParams fRetValue fValues
( ) type expr
( ) type expr
( ) type expr
( ) type expr
Int value
Int value
Int value
Int value
5
6
9
7

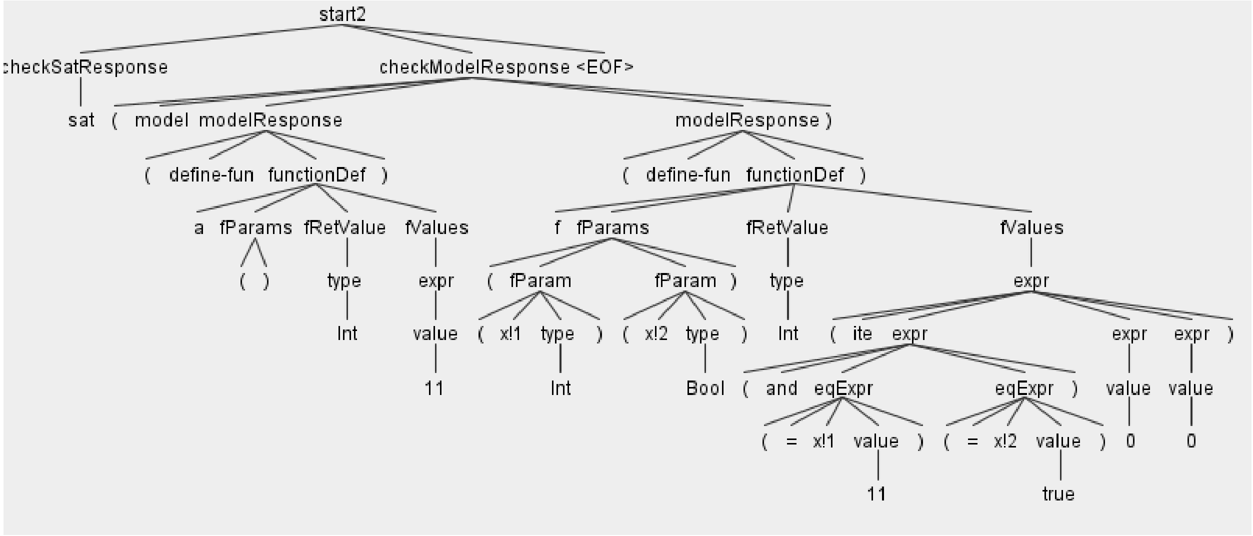
Test 2



```
sat
(model
  (define-fun a () Int
    11)
  (define-fun f ((x!1 Int) (x!2 Bool)) Int
    (ite (and (= x!1 11) (= x!2 true)) 0
      0))
)
```

Input:

Parse tree:


start2
checkSatResponse
checkModelResponse <EOF>
sat (model modelResponse
modelResponse)
(define-fun functionDef)
(define-fun functionDef)
a fParams fRetValue fValues
f fParams
fRetValue
fValues
( )
type
expr
(fParam
fParam)
type
expr
Int
value
(x!1 type)
(x!2 type)
Int
(ite expr
expr expr)
11
Int
Bool
(and eqExpr
eqExpr)
value value
(= x!1 value)
(= x!2 value)
0
0
11
true