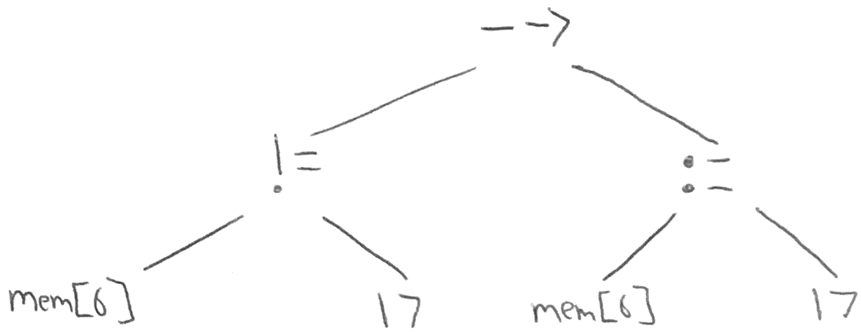
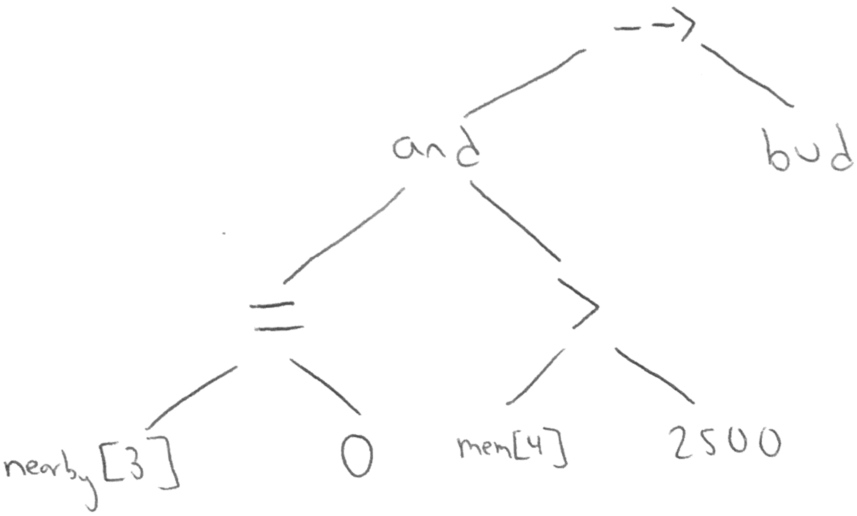
AST 1)

mem[6] != 17 --> mem[6] := 17;



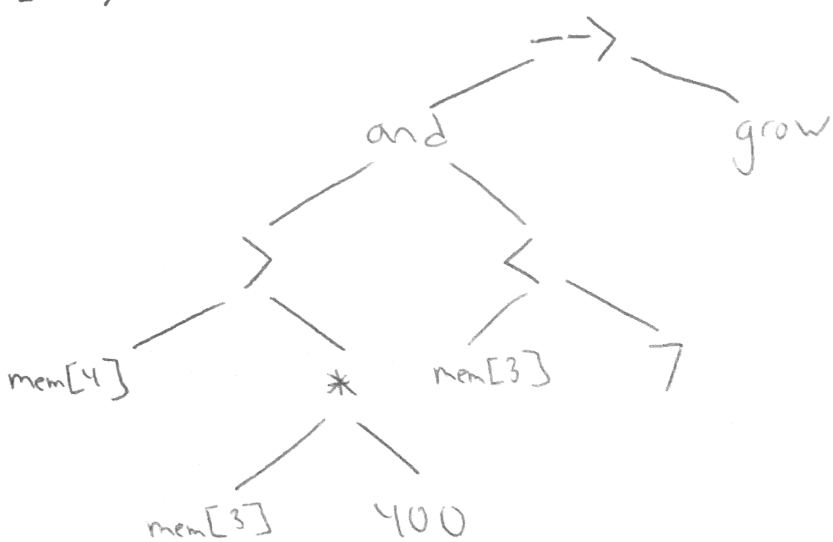
AST 2)

mem[3] = 0 and mem[4] > 2500 --> bud;



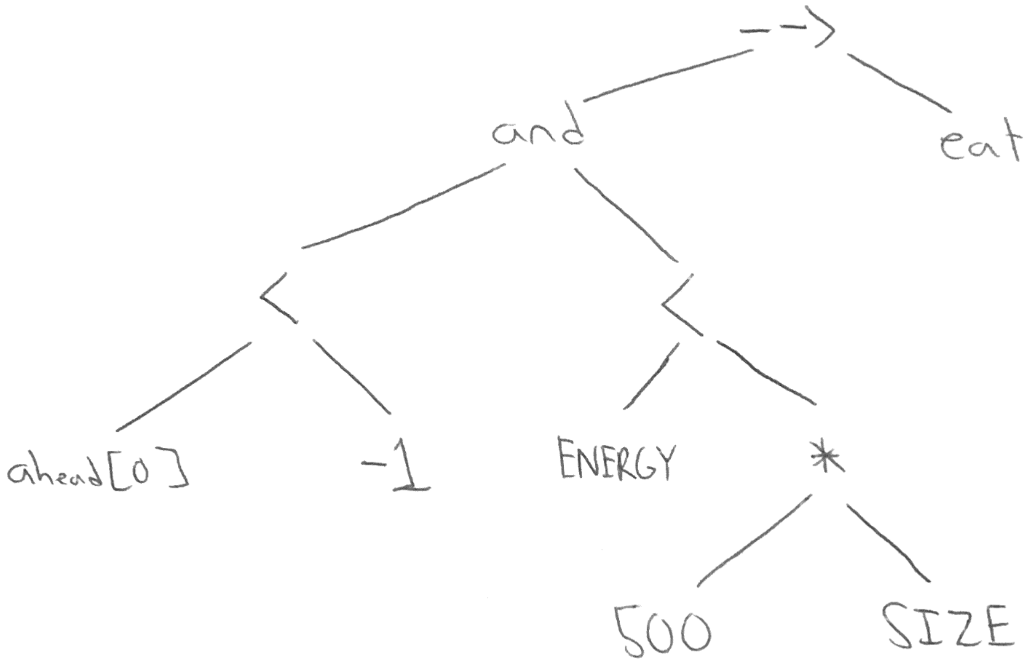
AST 3)

mem[4] > mem[3] * 400 and mem[3] < 7 --> grow



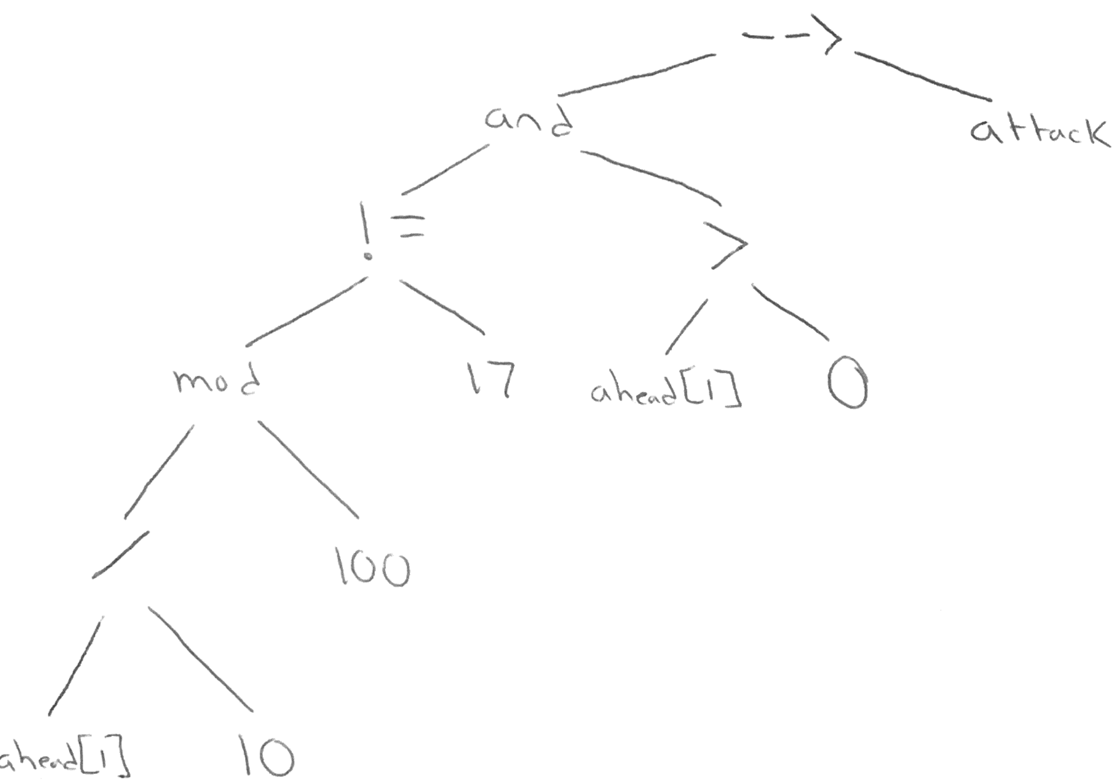
AST 4)

ahead[0]<-1 and ENERGY < 500 * SIZE --> eat



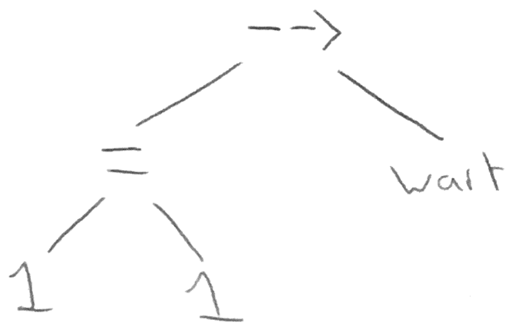
AST 5)

ahead[0]/10 mod 100 != 17 and ahead[1] > 0 --> attack



AST 6)

1=1 --> wait



Loop Invariant:

1. min & max contain the smallest & largest values (respectively) in the subarray [0, i).
2. a.length is even
3. i is an even integer within the range 0 <= i <= a.length

Establishment:

Initially, part 1 of the loop invariant is trivially true. Since no elements in the array have been encountered yet, max is set to MIN_VALUE and min is set to MAX_VALUE. Part 2 of the loop invariant is guaranteed by the precondition because a must be of even length. Part 3 of the loop invariant is initially true because i is initialized to 0 which satisfies 0 <= i <= a.length.

Preservation:

With each iteration of the loop, the length of a remains constant so part 2 of the invariant is preserved. Since i increments by two and starts on an even number i cannot become odd and since the loop terminates when i >= a.length-1 (meaning while the loop runs i<a.length-1) i remains within the range 0 <= i <= a.length thereby preserving clause 3. Finally, within each iteration of the loop the values x & y are assigned to the smaller & larger values (respectively) of the pair defined by a[i] and a[i+1]. The value of min is only changed when x<min and the value of max is only changed when y>max so clause 1 must be preserved because by the end of each iteration min and max contain the smallest and largest values encountered so far.

Postcondition:

Once again, since the length of a never changes, clause 2 of the loop invariant must be true after the loop terminates. Since the loop terminates when i>=a.length-1 and i increments by 2 with each iteration, on the last valid iteration i would have to be (a.length-1)-2 or (a.length-1)-1. Since (a.length-1)-2 is an odd number, which cannot be contained in i as previously proven, the last valid value of i has to be (a.length-1)-1 --> a.length-2. This means that the final value of i is (a.length-2)+2 or, a.length. Since i=a.length satisfies 0 <= i <= a.length clause 3 holds in the postcondition. Finally, since min & max are only updated when smaller and larger values (respectively) are found in the array, by the time the loop has iterated over every single pair of elements (which is guaranteed to be all of the elements because a.length is even) min and max must contain the minimum and maximum values found in the whole array which proves clause 1.