CptS 355
Homework 1
Solutions

1. Sebesta, Chapter 3, problem 3.
   Rewrite BNF of Example 3.4 to give + precedence over * and force + to be right associative.

Solution:

To give + precedence over * we change the grammar in this way,

<assign> ? <id> = <expr>

<id> ? A | B | C

<expr> ? <expr> * <term> | <term>

<term> ? <term> + <factor> | <factor>

<factor> ? (<expr>) | id

To force + to be right associative we change the grammar in this way,

<assign> ? <id> = <expr>

<id> ? A | B | C

<expr> ? <expr> * <term> | <term>

<term> ? **<factor> + <term>** | <factor>

<factor> ? (<expr>) | id

2. Write a grammar for the language over the alphabet {a, b} consisting of strings that read the same backward or forward (Palindromes). For example the strings a, aa, aba, abba, baab, abaaba are all in the language but the strings abb, ab, aabbb are not.

Solution:

Grammar for Palindromes

<S> ? a<S>a |
b<S>b |
a |
b |
epsilon

3. Draw parse trees for the sentences abaaaba, bbabb as derived from your grammar of Problem 2.

Solution:

The parse trees for the sentences abaaaba and bbabb are shown below in Figure 3.1 and Figure 3.2 respectively.

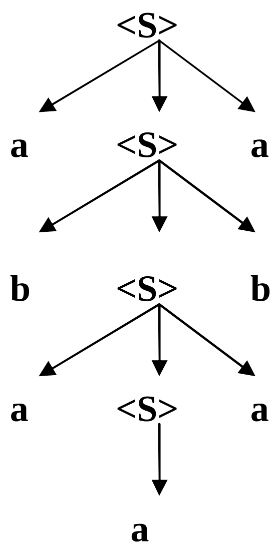

Figure 3.1

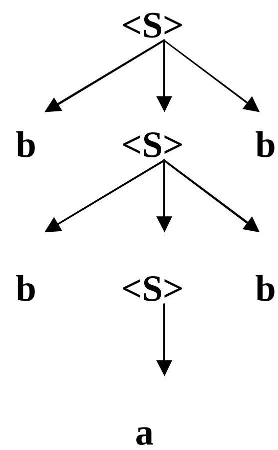

Figure 3.2

4. Sebesta, Chapter 3, problem 19.

Solution:
a. The assignment statement and the postcondition are

a = 2 * (b - 1) - 1 {a > 0}

Applying the assignment axiom we have the weakest precondition to be

{2 * (b - 1) - 1 > 0}
{2 * b - 3 > 0}
{b > 3/2 }

b. The assignment statement and the postcondition are

b = (c + 10) / 3 {b > 6}

Applying the assignment axiom we have the weakest precondition to be

{(c + 10) / 3 > 6}
{c + 10 > 18}
{c > 8}

c. The assignment statement and the postcondition are

a = a + 2 * b - 1 {a > 1}

Applying the assignment axiom we have the weakest precondition to be

{a + 2 * b - 1 > 1}
{a + 2 * b > 2}
{2 * b > 2 - a}
{b > (2 - a)/2}
{b > 1 - (a/2)}

d. The assignment statement and the postcondition are

x = 2 * y + x - 1 {x > 11}

Applying the assignment axiom we have the weakest precondition to be

{2 * y + x - 1 > 11}
{2 * y + x > 12}
{2 * y > 12 - x}
{y > 6 - (x/2)}

5. Sebesta, Chapter 3, problem 20

Solution:
a.) The assignment statements and their postconditions are

a = 2 * b + 1;
b = a - 3
{b < 0}

We have to apply the assignment axiom to each statement separately here.
Applying the assignment axiom first to the second assignment statement we get

{a - 3 < 0}
{a < 3}

Therefore {a < 3} is the weakest precondition for the second statement.
It becomes the post condition for the first assignment statement.

Hence,
{2 * b + 1 < 3}
{2 * b < 2}

{b < 1}

Therefore {b < 1} is the weakest precondition of the two statements.

b.) The assignment statements and their postconditions are

a = 3 * (2 * b + a);
b = 2 * a - 1
{b > 5}

Applying the assignment axiom first to the second assignment statement we get

{2 * a - 1 > 5}
{2 * a > 6}
{a > 3}
Therefore {a > 3} is the weakest precondition for the second statement.
It then becomes the post condition for the first assignment statement.

Hence,
{3 * (2 * b + a) > 3}
{2 * b + a > 1}
{a > 1 - 2 * b}

Therefore {a > 1 - 2 * b} is the weakest precondition of the two statements.

6. Consider the following program and postcondition:

```
i = n;
product = 1;
while i>k do
   product = product * i;
   i = i-1;
   end
{product = (n!/k!)}
```

You may assume that all values are integers.

a. Determine an invariant, I, for the while statement.
b. Determine a precondition, P, for the program as a whole
c. Show that P => wp(i=n; product=1, {I})
d. Show that (I and i>k) => wp(product = product*i; i=i-1, {I})
e. Show that (I and not i>k) => product = n!/k!
f. Argue informally that the loop terminates when the program starts in a state satisfying your P

Solution:

a. The invariant I for the while statement

The while body computes the product in order of last multiplication first until the value of ‘i’ is not bigger than k. The value of ‘i’ within the while body is progressively decreased by 1 from an initial value of ‘n’ for each iteration. Therefore the invariant I for the while statement is

I = (product = (i + 1) * (i + 2) * … * (n – 1) * n) AND (i >= k)

b. Precondition, P, for the program as a whole

Precondition P for the whole program is **(n >= k)** because the value of ‘i’ and hence the value of ‘n’ has to be initially greater than k. We assume that the value of k is always greater than zero.

c. Show that P => wp(i=n; product=1, {I})

The precondition, P, for the program as a whole is obtained by making I as the postcondition for the statements 1 & 2 as shown below

```
{P}
1. i = n;
2. product = 1;
{I}
```

Applying the assignment axiom first to the second assignment statement we get the weakest precondition to be

(1 = (i + 1) * (i + 2) * … * (n – 1) * n) AND (i >= k))

Taking the above expression to be the postcondition for the first assignment statement we get P to be

(1 = (n + 1) * … * n) AND (n >= k))

The left operand of the AND operator is true (a product of no terms is conventionally taken to equal 1) hence the precondition for the assignment statements 1 & 2 (precondition P) is

P = (n >= k)

d. Show that (I and i>k) => wp(product = product*i; i=i-1, {I})

(I and i > k) → (product = (i + 1) * (i + 2) * … * n) AND (i >= k) AND (i > k)
→ (product = (i + 1) * (i + 2) * … * n) AND (i > k)

wp(product = product*i; i=i-1, {I})
→ wp(product = product * i; (product = i * (i + 1) * … * n) AND (i - 1 >= k))
→ (product * i = i * (i + 1) * … * n) AND (i - 1 >= k))
→ (product = (i + 1) * (i + 2) * … * n) AND (i - 1 >= k)
→ (product = (i + 1) * (i + 2) * … * n) AND (i >= k + 1)
→ (product = (i + 1) * (i + 2) * … * n) AND (i > k)
→ (I and i > k)

e. Show that (I and not i>k) => product = n!/k!

(I and not i > k)

→ (I and i <= k)

→ (product = (i + 1) * (i + 2) * … * n) AND (i >= k) AND (i <= k)

→ (product = (i + 1) * (i + 2) * … * n) AND (i = k)

→ (product = (k + 1) * (k + 2) * … * n)

→ (product = n!/k!)

f. Argue informally that the loop terminates when the program starts in a state satisfying your P

When the program starts in a state satisfying P observe that the value of ‘i’ will initially be greater than or equal to k. In each loop iteration the value of ‘i’ is decremented by 1. Therefore the however big the initial value of ‘i’ its value eventually will be equal to k. The loop terminates when the value of ‘i’ is equal to k. Hence the loop will terminate.