Programming Languages: Solutions to Homework 1

B. Mishra

October 9 1995

Problem 1. [Problems 1.3 (pp. 21), 2.8 (pp. 60), 2.9 (pp. 60) in Sethi: Programming Languages—Concepts and Constructs.]

Solution.
1.3.(a)

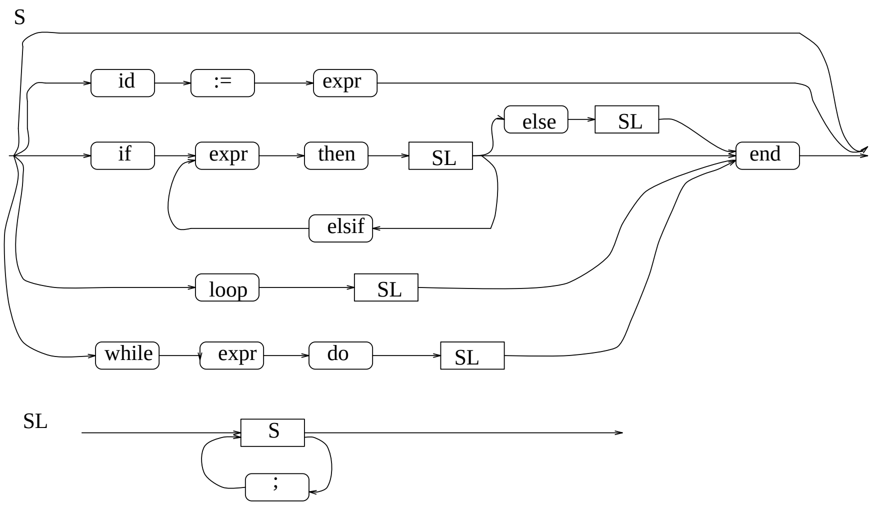
```
<s> ::= <empty>
       |  id := expr
       |  if expr then <sl> <ei> <el> end
       |  loop <sl> end
       |  while expr do <sl> end

<ei> ::= <empty> |  <ei> elsif expr then <sl>

<el> ::= <empty> | else <sl>

<sl> ::= <s>  | <s> ; <sl>
```

(b)



2.8
a) 4 b) 25 c) 25 d) 16
2.9.(a)

```
fun member(k, nil) = false |
    member(k, cons(j, y)) =
               if k = j then true else member(k, y);
```

b)

```
fun adjacent(nil) = false |
    adjacent(cons(x, nil)) = false |
    adjacent(cons(x, cons(y, z))) =
              if x = y then true else adjacent(cons(y,z));
```

c)

```
fun min(cons(i,nil)) = i |
    min(cons(i, cons(j, x))) =
            if i < j then min(cons(i, x)) else min(cons(j, x));
```

d)

```
fun less(k, nil) = nil |
    less(k, cons(j, x)) =
           if j < k then cons(j, less(k, x)) else less(k, x);
```

Problem 2. *What do following* `C` *expressions evaluate to?*

1. `a---b`

2. `a+++++b`

3.
```
if (x[i] > big);
  big = x[i];
```

4.
```
if (n<3)
  return
my.date = x[0];
my.time = x[1];
my.place = x[2];
```

5.
```
while (c =getc(in) != EOF)
   putc(c,out);
```

The intention is to copy a file of characters from `in` *to a new file* `out`. *Hint: Instead the file* `out` *will contain a string of bytes, each of binary value 1.*

Solution. According to Kernighan and Ritchie, "The C Programming Language," 2nd edition (K&R), p. 192, "If the input stream has been separated into tokens up to a given character, the next token is the longest string of characters that could constitute a token."
a) `a---b` evaluates as `(a--) - b`
b) `a+++++b` should evaluate as `((a++)++) + b`; but K&R says, on p. 203, that for both postfix and prefix incrementation, "The operand must be an l-value...The result is not an l-value." Therefore, this expression is illegal and will not compile.
c) `big = x[i];` is always evaluated, the value is `x[i]`, and `big` is set to that value.
d) If `n<3` then set `my.date` to `x[0]` and return that value; if `n>=3` then execute the last two statements.
e) This is equivalent to `while (c = (getc(in) != EOF)) putc(c,out);` The out stream will have as many ones written to it as there are characters remaining in the in stream.

Problem 3. *A simple string-searching program.*
(a) Here are a PASCAL type declaration and function header:

```
type string10 = array[1..10] of char;
function index(s: string10; c: char): integer;
```

The function `index` *is supposed to search the string* `s` *for the leftmost occurrence of the character* `c`. *If* `c` *is found the index of the leftmost occurrence is returned; otherwise* `0` *is returned. Examples:*

```
index('Hello, Bud', 'l')
   returns 3
index('Hi, Thomas', 'l')
   returns 0
```

Write the entire definition of the function index in PASCAL.
(b) Why is not the call `index('Huh?', 'u')` *legal?*
(c) Write a version of `index` *that can effectively take a string of any length and search it for a character. Invent a valid PASCAL data type* `VaryingString` *and use it as the first argument of a new version of the function* `index`. *Address the question (b) for your new function.*

Solution. a)

```
function index(s: string10; c: char): integer;
var i: integer;
begin
  i := 1;
  while ((s[i] <> c) and (i < 10)) do
    i := i+1;
```

```
    if (s[i] = c) then
      index := i
    else
      index := 0;
  end;
```

b) ´Huh?´ is not of type `string10` (i.e., `array[1..10] of char`).
c)

```
  type VaryingString = ^node;
       node = record
                c: char;
                next: VaryingString;
              end;
  function index(s: VaryingString; c: char): integer;
  var i: integer;
      p: VaryingString;
  begin
    i := 1;
    p := s;
    index := 0;
    while (p <> nil) do
      if (p^.c = c) then begin
        index := i;
        p := nil;
      end
      else begin
        i := i + 1;
        p := p^.next;
      end;  {if}
  end; {index}
```

The call in part (b) still won't work, since ´Huh?´ isn't type `VaryingString`. Instead the function will have to be called with `s` pointing to a linked list containing, `H, u, h,` and `?`.