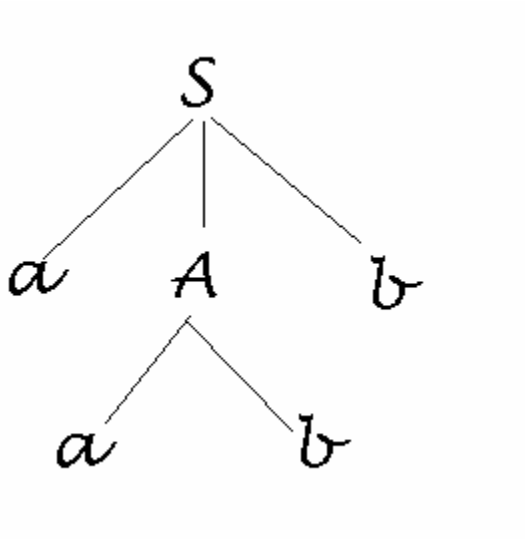
CS441 Fall 04 Internet Section

Solutions for hw #2

Chapter 3
Q14. Draw parse trees for the sentences aabb and aaaabbbb, as derived from the grammar of Problem 13.

Grammar is:
S -> aAb

A->aAb/ϕ



note: S stands for start.
a,b,c,d… are terminal symbols
A, B ,… etc are intermediate symbols.
ϕ denotes NULL element.

Q16. Convert the BNF of Example 3.3 to EBNF

EBNF : <assign> → <id> = <expr>

<id> → A | B | C

<expr> → <expr> (+ | -) <expr>

| (<expr>)

| <id>

Q18. Using the virtual machine instructions given in Section 3.5.1.1, given an operational semantic definition semantic definition of the following:

(a) (Pascal **`repeat`**) We assume that the logic expression is a single relational expression.

```
        loop:   ...
                ...
                if <relational_expression> goto out
                goto loop
        out:    ...
```

(b) (Ada **`for`**) **`for`** `I` **`in`** `first` **`..`** `last` **`loop`**

```
                I = first
    loop:       if I < last goto out
                ...
                I = I + 1
                goto loop
     out:       ...
```

(c) (Fortran Do)

```
                K = start
        loop:   if K > end goto out
                ...
                K = K + step
                goto loop
```

```
        out:    ...
```

(e) (C `for`) **for** (expr1; expr2; expr3) ...

```
                evaluate(expr1)
        loop:   control = evaluate(expr2)
                if control == 0 goto out
                ...
                evaluate(expr3)
                goto loop
        out:    ...
```

Q19. Compute the weakest precondition for each of the following assignment statements and postconditions:

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

2 * (b - 1) - 1 > 0

2 * b - 2 - 1 > 0

2 * b > 3

b > 3 / 2

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

(c + 10) / 3 > 6

c + 10 > 18

c > 8

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

a + 2 * b - 1 > 1

2 * b > 2 - a

b > 1 - a / 2

(d) $x = 2 * y + x - 1$ $\{x > 11\}$

$2 * y + x - 1 > 11$

$2 * y + x > 12$

Q21. Write a denotational semantics mapping function for the following statements:

M_{pf}(for var in init_expr .. final_expr loop L end loop, s) $\triangleq$

if VARMAP(i, s) = **undef** for var or some i in init_expr or final_expr

then **error**

else if M_e(init_expr, s) > M_e(final_expr, s)

then s

else M_l(while init_expr - 1 <= final_expr do L, M_a(var := init_expr + 1, s))

Chapter 4

Q1. Perform the pairwise disjointness test for the following grammar rules.

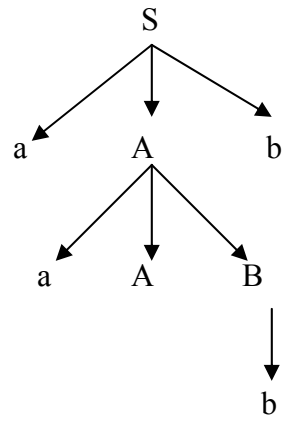
(a) FIRST(aB) = {a}, FIRST(b) = {b}, FIRST(cBB) = {c}, Passes the test

(b) FIRST(aB) = {a}, FIRST(bA) = {b}, FIRST(aBb) = {a}, Fails the test

(c) FIRST(aaA) = {a}, FIRST(b) = {b}, FIRST(caB) = {c}, Passes the test

Q3. Given the following grammar and the right sentential form, draw a parse tree and show the phrases and simple phrases, as well as the handle.

(a) aaAbb

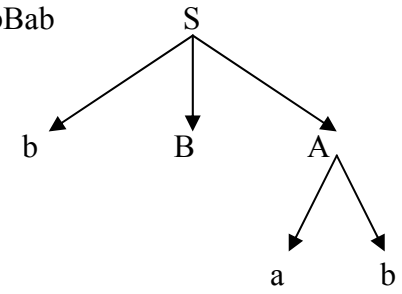


Phrases: aaAbb, aAb, b

Simple phrases: b

Handle: b

(b) bBab



Phrases: bBab, ab

Simple phrases: ab

Handle: ab

*Q5. Show a complete parse, including the parse stack contents, input string, and action for the string id * (id + id), using the grammar and parse table in Section 4.5.3.*

Stack	*Input*	*Action*
0	id * (id + id) $	Shift 5
0id5	* (id + id) $	Reduce 6 (Use GOTO[0, F])
0F3	* (id + id) $	Reduce 4 (Use GOTO[0, T])
0T2	* (id + id) $	Reduce 2 (Use GOTO[0, E])
0T2*7	(id + id) $	Shift 7
0T2*7(4	id + id) $	Shift 4
0T2*7(4id5	+ id) $	Shift 5
0T2*7(4F3	+ id) $	Reduce 6 (Use GOTO[4, F])
0T2*7(4T2	+ id) $	Reduce 4 (Use GOTO[4, T])
0T2*7(4E8	+ id) $	Reduce 2 (Use GOTO[4, E])
0T2*7(4E8+6	id) $	Shift 6
0T2*7(4E8+6id5	) $	Shift 5

0T2*7(4E8+6F3	) $	Reduce 6 (Use GOTO[6, F])
0T2*7(4E8+6T9	) $	Reduce 4 (Use GOTO[6, T])
0T2*7(4E8	) $	Reduce 1 (Use GOTO[4, E])
0T2*7(4E8)11	$	Shift 11
0T2*7F10	$	Reduce 5 (Use GOTO[7, F])
0T2	$	Reduce 5 (Use GOTO[0, T])
0E1	$	Reduce 2 (Use GOTO[0, E])
--ACCEPT--		

Chapter 5

Q2. Some programming languages are type less. What are the obvious advantages and disadvantages of having no types in a language?

The advantage of a type less language is flexibility; any variable can be used for any type values. The disadvantage is poor reliability due to the ease with which type errors can be made, coupled with the impossibility of type checking detecting them.

Q8. a. Assuming static scoping, which declaration of x is the correct one for a reference to x in the following:

(a) i. `Sub1`

ii. `Sub1`

iii. `Main`

(b) i. `Sub1`

ii. `Sub1`

iii. `Sub1`

Q10. List all the variables, long with the program units where they are declared, that are visible in the bodies of Sub1, Sub2, and Sub3, assuming static scoping is used.

Variable	Where Declared
In `Sub1`:	
`A`	`Sub1`
`Y`	`Sub1`

```
     Z    Sub1
     X    Main
```

In `Sub2`:

```
     A    Sub2
     B    Sub2
     Z    Sub2
     Y    Sub1
     X    Main
```

In `Sub3`:

```
     A    Sub3
     X    Sub3
     W    Sub3
     Y    Main
     Z    Main
```

Q13. Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last function called? Include with each visible variable the name of the function in which it was defined.

(a)
```
d, e, f    fun3
c          fun2
b          fun1
a          main
```

(b)
```
d, e, f    fun3
b, c       fun1
a          main
```

(c)
```
b, c, d    fun1
e, f       fun3
a          main
```

(d)
```
b, c, d    fun1
e, f       fun3
a          main
```

(e)
```
c, d, e    fun2
f          fun3
```

```
      b          fun1
      a          main
(f)   b, c, d    fun1
      e          fun2
      f          fun3
      a          main
```

Chapter 6

Q2. Why does a decimal value waste memory space?

Integer values stored in decimal waste storage in binary memory computers, simply as a result of the fact that it takes four binary bits to store a single decimal digit, but those four bits are capable of storing 16 different values. Therefore, the ability to store six out of every 16 possible values is wasted. Numeric values can be stored efficiently on binary memory computers only in number bases that are multiples of 2. If humans had developed a number of fingers that was a power of 2, these kinds of problems would not occur.

Q6. What disadvantages are there in implicit dereferencing of pointers, but only in certain contexts? For example, consider the implicit dereference of a pointer to a record in Ada when it is used to reference a record field.

When implicit de-referencing of pointers occurs only in certain contexts, it makes the language slightly less orthogonal. The context of the reference to the pointer determines its meaning. This detracts from the readability of the language and makes it slightly more difficult to learn.

Q8. The union in C and C++ are separate from the records of those languages, rather than combined as they are in Ada. What are the advantages and disadvantages to these two choices?

The advantage of having a separate construct for unions is that it clearly shows that unions are different from records. The disadvantages are that it requires an additional reserved word and that unions are often separately defined but included in records, thereby complicating the program that uses them.

Q9. Multidimensional arrays can be stored in row major order, as in C++, or in column major order, as in Fortran. Develop the access functions for both of these arrangements for three-dimensional arrays.

. Let the subscript ranges of the three dimensions be named `min(1)`, `min(2)`, `min(3)`, `max(1)`, `max(2)`, and `max(3)`. Let the sizes of the subscript ranges be `size(1)`, `size(2)`, and `size(3)`. Assume the element size is 1.

Row Major Order:

location(a[i,j,k]) = (address of a[min(1),min(2),min(3)])

```
     +((i-min(1))*size(3) + (j-min(2)))*size(2) + (k-min(3))
```

Column Major Order:

location(a[i,j,k]) = (address of a[min(1),min(2),min(3)])

```
     +((k-min(3))*size(1) + (j-min(2)))*size(2) + (i-min(1))
```

Q10. In the Burroughs Extended ALGOL language, matrixes are stored as a single-dimensioned array of pointers to the rows of the matrix, which are treated as single-dimensioned arrays of values. What are the advantages and disadvantages of such a scheme?

The advantage of this scheme is that accesses that are done in order of the rows can be made very fast; once the pointer to a row is gotten, all of the elements of the row can be fetched very quickly. If, however, the elements of a matrix must be accessed in column order, these accesses will be much slower; every access requires the fetch of a row pointer and an address computation from there. Note that this access technique was devised to allow multidimensional array rows to be segments in a virtual storage management technique. Using this method, multidimensional arrays could be stored and manipulated that are much larger than the physical memory of the computer