

60-100 LECTURE 17 d

CLASS TEST # 2 – November 13th 2010

OPERATIONS ON DATA

(1) What is the type of the program `g` defined as follows:

```
g x y [] = 1 : x
g x y [a] = x ++ y
g x y (a : as) = a
```

(2) Write Miranda programs to implement the following relational algebra operator:

```
join_third_of_4_with second_of_2
```

(3) $p = \{(6, 'w'), (9, 'y'), (4, 'w')\}$
 $q = \{('w', 5), ('w', 7), ('y', 2)\}$

What is the result of the following expression
(show all intermediate results):

```
project_third_of_3  
(join_second_of_2_with_first_of_2  
  p (select_first_of_2 'w' q))
```

RECURSION

(4) Let f be defined as follows:

f [] n = [n]

f y 0 = y

f(x:xs) n = f xs (n-1) ++ f xs (n+1)

what is the value of the following application of the program f:

f [1, 2, 3] 5

(5) Use **recursion** to define a program `p5` which takes one number `n` as input and which returns the value `n mod 10`. That is, it returns the remainder after `n` has been divided by 10. For example:

`p5 32 => 2`

`p5 425 => 5`

You MUST use recursion, and **MUST NOT** use the `mod` or `div` or `/` operators.

(6) Use **recursion** to define a program `p6` which takes two lists as input and which appends the second list to the end of the first list. For example:

`p6 [2,3,3] [4,8,9] => [2,3,3,4,8,9]`

You **MUST** not use the `++` operator.

(7) Let the grammar G be defined as follows:

```
(terminals      = { [, ], * }
start symbol    = string
non-terms      = { string, single,
                  double, stars }

production rules
= { string ::= stars
    | double double
    | double single
  single ::= [ stars ]
  double ::= [ string ] [ stars ]
  stars  ::= *
          | * stars
```

Derive [*] [*] [**]

(8) Show the parse tree (i.e. the syntax tree) for the expression

[*] [*] [*] [* *]

(9) Construct a grammar for the language L:

{ "20.23" ,
"12.00power2" ,
"04.50power3" , etc.

(10) Construct an attribute grammar for the language L so that:

{ "20.23"	has value	20.23
"12.00power2"	has value	144
"04.50power3"	has value	91.125

IF TIME

Write recursive programs to do the following:

- 1) Append 2 lists (i.e. $n ++ m$)
- 2) Check to see if an element e is a member of a list n
- 3) Subtract a list n from a list m (i.e. $m - - n$)