Individual Assignment #5 to be submitted at the group labs on 23rd or 24th October 2014

CLASS TEST#2 2006

OPERATIONS ON DATA

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

```
g   ys    0  m  = 2
g   []    n  m  = 3
g  (x:xs) n  m  = g m (n – 3) (x:xs)  + g [x] (n – 3) m
```

(2) Definethe relational algebra operator `select_second_of_3`

(3) Let
```
p = {('w', 5), ('y', 6), ('z', 5)}
q = {('y', 3), ('a', 4)}
```

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

```
project_third_of_3
      (join_first_of_2_with_first_of_2  q (select_second_of_2 6 p))
```

RECURSION

(4) Let h be defined as follows:

```
h   ys    0   = 2
h   []    n   = 3
h  (x:xs) n   = h (x:xs) (n – 3) + h [x] (n – 3)
```

what is the value of the following application of the program `g`:

`g  [4, 5]` 9

(5) Use recursion to define a program `p5` which takes two numbers as input and which returns `True` if the first number is exactly divisible by the second number and `False` otherwise. For example:

```
p5 32 8 => True
p5 32 5 => False
```

You MUST use recursion, and MUST NOT use the `mod or div` operator.

(6) Use recursion to define a program called `p6` which takes a list of numbers as input and which returns the largest number in the list as output. For example:

```
p6 [4, 2, 3, 6, 1]        => 6
p6 [8, 9, 23, 4, 12]      => 23
```

CLASS TEST # 2 – 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) Let
```
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 dived 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

60-100 CLASS TEST # 2 – 2011

OPERATIONS ON DATA

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

```
g   []     n  m  = []
g   [x]    n  m  = [n]
g  (x:xs)  n  m  = g (m:xs) n (m – 1) ++ g xs n (m – 2)
```

(2) Write a Miranda program for the following relational-algebra operator:

```
joinsecond_of_3_with_first_of_2
```

(3) Let `p = {('y', 5), ('y', 6), ('z', 5)}`
`q = {('y', 33, 5), ('z', 44, 7), ('z', 44, 8)}`

What is the result of the following relational algebraic expression:

```
select_second_of_3  5
       (join_first_of_2_with_first_of_2  p
                   (project_first_and_second_of_3 q))
```

RECURSION

(4) Let h be defined as follows:

```
h   []    n   = [n]
h   xs    0   = [2]
h  (x:xs) n   = h xs (n – 1) ++ h [x] (n – 1)
```

what is the value of the following application of the program: `h [4, 5] 3`

(5) Use recursion to define a program `p5` which takes two numbers `n` and `m` as input and which returns the value `n ^ m.` For example:

```
p5 2 3  => 8
p5 3 4  => 81
```

You MUST use recursion, and MUST NOT use the `^` operator.

(6) Use recursion to define a program called `p6` which takes two lists `s` and `t` as input and which returns the number of elements in `s` that are also in `t`. For example:

```
p6 [2,5,4,3] [4,2,3,6,1] => 3 (because 2,4 and 3 are in both lists)
p6 [3,5,6]   [8,9,3,4,2] => 1 (because 3 is in both lists)
```

Assume that the program `member` is available, such that `member p e` returns `True` if `e` is in the list `p` and `False` otherwise. You cannot use the operator `--`.