Matthew O'Connor Student Number 50000250 Homework 4 Cse 305

Q1.) Upon testing the code in Java my output was:

a) I are the same

b) I are the same

Q2.) Upon Java testing:

The size of keyvalueue is 2

The count is 0

Q3.)

A.) 3, 5

B.) 6, 10

C.) 6, 10

Q4.) A multiple selector can be built from two-way selectors and go to statements.,

However the resulting structures are just bad in general, being unwieldy inefficient, and difficult to use at all(writing and reading).

Q5.)

a) (left->right) sum1 is 46; sum2 is 48

b) (right-> left) sum1 is 48; sum2 is 46

Q6.)

addition(X,[],[X]).

addition(X,[Y|Tail],[Y|Tail2]) :- addition(X,Tail,Tail2).

Q7.)

```
del(a, [a|B], B).
del(a, [B, C|D], [B|E]) :- delete(A, [C|D], E).
```

Q8.)

```
quick([], []).
quick([HEAD | TAIL], SORTED) :- partify(HEAD, TAIL, LEFT, RIGHT),
  quick(LEFT, SORTEDL),
  quick(RIGHT, SORTEDR),
  appendify(SORTEDL, [HEAD | SORTEDR], SORTED).
partify(pivot_point, [], [], []).
partify(pivot_point, [HEAD | TAIL], [HEAD | LEFT], RIGHT) :- HEAD @=< pivot_point,
 partify(pivot_point, TAIL, LEFT, RIGHT).
partify(pivot_point, [HEAD | TAIL], LEFT, [HEAD | RIGHT]) :- HEAD @> pivot_point,
partify(pivot_point, TAIL, LEFT, RIGHT).
appendify([], LIST, LIST).
appendify([HEAD | LIST1], LIST2, [HEAD | LIST3]) :- appendify(LIST1, LIST2, LIST3).
```

Q9.)

```
flattenize(List, flattenizeed):-
  flattenize(List, [], flattenizeed).

flattenize([], flattenizeed, flattenizeed).
flattenize([Item|Tail], L, flattenizeed):-
  flattenize(Item, L1, flattenizeed),
  flattenize(Tail, L, L1).
```

flattenize(Item, flattenizeed, [Item|flattenizeed]);

Q10.) this is what I get by checking the system

a) 'a -> 'b, 'b -> 'a.

b) int -> 'a.

c) 'a list * ('a -> 'b) -> 'b list.

11.)fun reduce1 (nil,FUNC) = raise EmptyList

reduce1 ([a].FUNC) = a (* rem I want F to take two args *)

reduce1 (FUNC,x::xs) = F(x,reduce(F,xs)) ;

Q12.) the result of reduce(op -, L) would be the list but negative. so (-a1,-a2 ect).

Q13.)

a.) Reduce(Or,List[])

b.)

```
fun concatz (x:strings):string =
  case y of []
    [] => ""      // looks for empties
  | h::t=> h ^ (concatz t);
```

Q14.)

a.)

```
   fun filtrate(Y,nil) = nil;
 filtrate(Y,x) =
 if Y(x) then x::filtrate(Y,xz)
else filtrateY,xz);
x++;
if x<3 then stop;
```

Q15.) reduce(map(x,y),z);

Q16.)fun leapz(y:int)= if x mod 400 = 0 then true

else if x mod 100 = 0 then false

else if x mod 4 = 0 then true else false;

Q17.)

```
fun quick nil = nil
|   quick (pivot_point :: rest) =
   let
 fun split(nil) = (nil,nil)
   split(y :: yz) =
 let
  value (below, above) = split(yz)
   in
  if y < pivot_point then (y:: below, above)
   else (below, y :: above)
    end;
  value (below, above) = split(rest)
   in
  quick below quick above
   end;
```