Assignment 4 – Functional Programming

By Shefali Anand
sanand22@asu.edu

Problem 1

Code:

```
(* sum of a List*)

fun sumList([]) = 0
| sumList(h::t) = h + sumList(t);
```

Sample Run:

```
sumList[22,4,7,1];
```

Soution:

```
- val sumList = fn : int list -> int
val it = 34 : int
-
```

Problem 2

Code:

```
(* Fibonacci *)

fun fib (pre, acc, n) =
   if n=1 then acc
   else fib (acc, pre+acc, n-1);

fun fibonacci(n) = fib (0,1,n);
```

Sample Run:

```
fibonacci(6);
```

Solution:

```
- val fib = fn : int * int * int -> int
val fibonacci = fn : int -> int
val it = 8 : int
-
```

Problem 3

Code:

```
(* Reverse List *)

fun rev (acc, []) = acc
| rev (acc, h::t) = rev ([h]@acc, t);

fun reverse(L) = rev ([], L);
```

Sample Run:

```
reverse([1,2,3,4,5]);
```

Solution:

```
- val rev = fn : 'a list * 'a list -> 'a list
val reverse = fn : 'a list -> 'a list
val it = [5,4,3,2,1] : int list
-
```

Problem 4

Code:

```
(* Rotate List *)

fun rot(r, h::t, L) =
  if r=0 then [h]@t@L
  else rot(r-1, t, L@[h]);

fun rotate(L,r) = rot(length(L)-(r mod length(L)),L,[]);
```

Sample Run 1:

```
rotate([1,2,3,4,5],3);
```

Solution 1:

```
val rot = fn : int * 'a list * 'a list -> 'a list
val rotate = fn : 'a list * int -> 'a list
val it = [3,4,5,1,2] : int list
-
```

Sample Run 2:

```
rotate([1,2,3,4,5],1);
```

Solution 2:

```
val rot = fn : int * 'a list * 'a list -> 'a list
val rotate = fn : 'a list * int -> 'a list
val it = [5,1,2,3,4] : int list
-
```

Sample Run 3:
rotate([1,2,3,4,5],8);

Solution 3:
```
val rot = fn : int * 'a list * 'a list -> 'a list
val rotate = fn : 'a list * int -> 'a list
val it = [3,4,5,1,2] : int list
-
```

Problem 5

Code:
```
(* Miles per Gallons *)

fun miles([])= 0
| miles((h,t)::T) = h+miles(T) ;

fun gallons([]) = real(0)
| gallons((h,t)::T) = t+gallons(T) ;

fun mpg(L) = [real(miles(L)), gallons(L), real(miles(L))/gallons(L)];
```

Sample Run:
mpg [(2,3.0), (5,4.4), (10,1.2)];

Solution:
```
- val miles = fn : (int * 'a) list -> int
val gallons = fn : ('a * real) list -> real
val mpg = fn : (int * real) list -> real list
val it = [17.0,8.6,1.97674418605] : real list
-
```

Problem 6

Code:
```
(* Merge Sort *)

fun split L =
  let
    val n = length(L) div 2
  in
    (List.take(L,n), List.drop(L,n))
  end;

fun merge([],L) = L
| merge(L,[]) = L
| merge( h1::t1 , h2::t2 ) =
   if h1<h2 then h1::merge(t1,h2::t2)
   else h2::merge(h1::t1,t2);

fun mergesort([]) = []
| mergesort(L) =
   if length(L)=1 then L
   else
   let
      val (first, second) = split(L)
      val f = mergesort(first)
      val s = mergesort(second)
   in
      merge(f,s)
   end;

fun msort(L) = mergesort(L);
```

Sample Run:

```
msort([2,5,3,4,1])
```

Solution:

```
 val split = fn : 'a list -> 'a list * 'a list
 val merge = fn : int list * int list -> int list
 val mergesort = fn : int list -> int list
 val msort = fn : int list -> int list
 = val it = [1,2,3,4,5] : int list
```

Problem 7

Code:

```
(* Tower of Honoi *)

fun move(1,s,c,d) = [(s,d)]
| move(x,s,c,d) = move(x-1, s,d,c)@move(1,s,c,d)@move(x-1,c,s,d);

fun hanoi(x,s,c,d) = move(x,s,c,d);
```

Sample Run:

```
hanoi(3,1,2,3);
```

Solution:

```
 - val move = fn : int * 'a * 'a * 'a -> ('a * 'a) list
 val hanoi = fn : int * 'a * 'a * 'a -> ('a * 'a) list
 val it = [(1,3),(1,2),(3,2),(1,3),(2,1),(2,3),(1,3)] : (int * int) list
 -
```
