Chapter 5

1. Write a function **cube** of type int->int that returns the cube of its parameter
   fun cube x = x * x * x;
   val cube = fn : int -> int

2. Write a function **cuber** of type real->real that returns the cube of its parameter
   fun cuber x:real = x * x * x;
   val cuber = fn : real -> real

3. Write a function **fourth** of type ‘a list -> ‘a that returns the fourth element of a list. Your function need not behave well on lists with less than four elements.
   fun fourth x = hd (tl (tl (tl x)));
   val fourth = fn : 'a list -> 'a

4. Write a function **min3** of type int * int * int -> int that returns the smallest of three integers.
   fun min3 (a,b,c) =
   = if a < b andalso a < c then a
   = else if b < c then b
   = else c;
   val min3 = fn : int * int * int -> int

5. Write a function **red3** of type ‘a * ‘b * ‘c -> ‘a * ‘c that converts a tuple with three elements into one with two elements by eliminating the second element.
   fun red3 (a,b,c) = (a,c);
   val red3 = fn : 'a * 'b * 'c -> 'a * 'c

6. Write a function **thirds** of type string -> char that returns the third character of a string. Your function need not behave well on strings with lengths less than 3.
   fun thirds s = hd(tl(tl(explode s)));
   val thirds = fn : string -> char

7. Write a function **cycle1** of type ‘a list -> ‘a list whose output list is same as the input list, but with the first element of the list moved to the end. For example, cycle1 [1,2,3,4] should return [2,3,4,1].
   fun cycle1 x = (tl x) @ [(hd x)];
   val cycle1 = fn : 'a list -> 'a list

8. Write a function **sort3** of type real * real * real -> real list that returns a sorted list of three real numbers.
   fun sort3 (a:real,b,c) = if a <= b andalso a <= c andalso b <= c then [a,b,c]
   = else if a <= b andalso a <= c andalso c <= b then [a,c,b]
   = else if b <= c andalso a <= c then [b,a,c]

```
= else if  b <= c andalso c <= a then [b,c,a]
= else [c,b,a];
val sort3 = fn : real * real * real -> real list
```

9. Write a function **del3** of type ‘a list -> ‘a list whose output list is same as the input list, but with the third element of the list deleted. Your function need not behave well on lists with lengths less than 3.
```
fun del3 x = hd x :: hd ( tl x )  :: tl( tl ( tl x ) ) ;
val del3 = fn : 'a list -> 'a list
```

10. Write a function **sqsum** of type int->int that takes a non-negative integer n and returns the sum of squares of all integers 0 through n. Your function need not behave well on inputs less than zero.
```
fun sqsum x = if x=0 then 0
= else ( x*x ) + sqsum (x-1);
val sqsum = fn : int -> int
```

11. Write a function **cycle** of type ‘a list * int -> ‘a list that takes a list and an integer as input and returns the same list, but with the first element cycled to the end of list n times. For example, cycle ([1,2,3,4,5,6],2) should return [3,4,5,6,1,2].
```
 fun cycle (x,y) = if y=0 then x
= else cycle (((tl x) @ [(hd x)]),(y-1));
val cycle = fn : 'a list * int -> 'a list
```

12. Write a function **pow** of type real * int -> real that raises a real number to an integer power. Your function need not behave well if the integer power is negative.
```
fun pow (x,y) = if y = 0 then 1.0
= else x*(pow(x,(y-1)));
val pow = fn : real * int -> real
```

13. Write a function **max** of type int list -> int that returns the largest element of a list of integers. Your function need not behave well if the list is empty. Write a helper function **maxhelper** that takes as a second parameter the largest element seen so far.
```
fun max x = maxhelper (tl x , hd x);
fun maxhelper (x,y) = if null x then y
= else if (hd x) > y then maxhelper(tl x, hd x)
= else maxhelper(tl x , y);
val maxhelper = fn : int list * int -> int
fun max x = maxhelper(tl x , hd x);
val max = fn : int list -> int
```

14. Write a function **isPrime of type int -> bool** that returns true iff its integer parameter is a prime number.

```
 fun divisor (m,n)  =  if n mod m = 0 then false
= else if (m*m) > n then true
= else divisor(m+1,n);
val divisor = fn : int * int -> bool
- fun isPrime n = divisor (2,n);
val isPrime = fn : int -> bool
```

15. Write a function **select** of type ‘a list * (‘a -> bool) -> ‘a list that takes a list and a function f as parameters. Your function f should be applied to each element of the list and should return only those elements of the original list for which f returned true.

```
 fun select (x, f:'a -> bool) = if null x then nil
= else if f((hd x))=true then (hd x) :: select ((tl x), f)
= else select ((tl x) , f);
val select = fn : 'a list * ('a -> bool) -> 'a list
```