OCaml examples

Disclaimer: These notes may be distributed outside this class only with the permission of the Instructor.

1.1 OCaml code examples

Listing 1: is vowel

```ocaml
let is_vowel c =
  c='a' || c='e' || c='o' || c = 'i' || c = 'u' ; ;
```

Listing 2: is vowel

```ocaml
let is_vowel c =
  c='a' || c='e' || c='o' || c = 'i' || c = 'u' ; ;
```

Listing 3: is vowel

```ocaml
let is_vowel c =
  match c with
  | 'a'->true
  | 'o'->true
  | 'u'>='e'
  | 'e'->true
  |->'i'->true ; ;
```

Listing 4: is vowel

```ocaml
let is_vowel c =
  match c with
  | 'a'|| 'o'|| 'u'|| 'e'|| 'i'->true
  |_.->false ; ;
```

Listing 5: isnil

```ocaml
let isnil list =
  match list with
  |[]->true
  |_.->false ; ;
```
Listing 6: length of a list

```
let rec length list =
  match list with
  [] -> 0
  | h :: t -> 1 + length t
```

Listing 7: reverse a list

```
let rec rev list =
  match list with
  [] -> []
  | h :: t -> rev t @ [h]
```

Listing 8: sum of a list of integers

```
let rec sum list =
  match list with
  [] -> 0
  | h :: t -> h + sum t
```

Listing 9: Append a list to another list

```
let rec append a b =
  match a with
  [] -> b
  | h :: t -> h :: append t b
```

Listing 10: A list of integer in a given range

```
let rec range a b =
  if a > b then []
  else a :: range (a + 1) b;;
```

Listing 11: range 5 10

```
let r = range 5 10;;
```

Listing 12: first integer of the list

```
let first l =
  match l with
  [] -> 0
  | h :: _ -> h;;
```
Listing 13: last integer of the list

```ocaml
let rec last l =
  match l with
  | [] -> 0
  | [x] -> x
  | h :: t -> last t
```

Listing 14: factorial

```ocaml
let rec fact n =
  if n = 0 then 1
  else n * fact (n - 1);
```

Listing 15: concat a list

```ocaml
let rec concat l =
  match l with
  | [] -> ""
  | h :: t -> h ^ concat t ; ;
```

Listing 16: map

```ocaml
let rec map f l =
  match l with
  | [] -> []
  | h :: t -> f h :: (map f t)
```

Listing 17: fold

```ocaml
let next (a, _)=a + 1;;
let fold (f, a, l) =
  match l with
  | [] -> a
  | (h :: t) -> fold (f, f (a, h), t); ;
```

Listing 18: reverse a list using fold

```ocaml
let prepend (a, x) = x :: a ; ;
let fold (prepend, [], [1;2;3;4;5;6;7]) ; ;
```

Listing 19: sum of a list

```ocaml
let sum list =
  fold ((fun (a,x) -> a + x), 0, list )
```

Listing 20: sum of a list

```ocaml
let sum list =
  let add (a, x)=a + x in
  fold (add, 0, list )
```
Listing 21: merge 2 lists

```ml
let rec merge l1 l2 =
  match l1 with
  | [] -> l2
  | a::t -> h::merge l2 t
```

Listing 22: insert an item to a sorted list

```ml
let rec insert x l =
  match l with
  | [] -> [x]
  | h::t -> if x < h then x::h::t
    else h::insert x t
```

Listing 23: insertion sort

```ml
let rec sort l =
  match l with
  | [] -> []
  | h::t -> print_int h; insert h (sort t)
```

1.1.1 Number to Word

Listing 24: Number to Word Conversion

```ml
(*
This program converts a number to the english word
15 => fifteen
123 => one hundred twenty three
*)

let get_ones x =
  match x with
  | 0 -> ""
  | 1 -> "one"
  | 2 -> "two"
  | 3 -> "three"
  | 4 -> "four"
  | 5 -> "five"
  | 6 -> "six"
  | 7 -> "seven"
  | 8 -> "eight"
  | 9 -> "nine"
  | 10 -> "ten"
  | 11 -> "eleven"
  | 12 -> "twelve"
  | 13 -> "thirteen"
  | 14 -> "fourteen"
  | 15 -> "fifteen"
```
let get_tens x =
  match x with
  | 16 -> "sixteen"
  | 17 -> "seventeen"
  | 18 -> "eighteen"
  | 19 -> "nineteen"
  | _ -> ""

let rec convert num =
  let aux (d, str) =
    let t1 = num / d in
    let t2 = num mod d in
    (convert t1) ^ str ^ (convert t2) in
    if num >= 1000000000 then
      aux (1000000000, "billion")
    else if num >= 100000 then
      aux (100000, "million")
    else if num >= 1000 then
      aux (1000, "thousand")
    else if num >= 100 then
      aux (100, "hundred")
    else if num >= 20 then
      let t1 = num / 10 in
      let t2 = num mod 10 in
      (get_tens t1) ^ " " ^ (convert t2)
    else
      get_ones num
  in
  aux 10 ""

let n = 30;;
print_int n;;
print_newline();;
print_string (convert n);;
print_newline();;
References

[OCaml from the very beginning]   JOHN WHITTINGTON Coherent Press