

OCaml examples

*Lecturer:***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

```
1 let is_vowel c=  
2     c='a' || c='e' || c='o' || c='i' || c='u'  
3 ;;
```

Listing 2: is vowel

```
1 let is_vowel c=  
2     c='a' || c='e' || c='o' || c='i' || c='u'  
3 ;;
```

Listing 3: is vowel

```
1 let is_vowel c=  
2     match c with  
3         | 'a' -> true  
4         | 'o' -> true  
5         | 'u' -> 'e'  
6         | 'e' -> true  
7         | _ -> 'i' -> true  
8 ;;
```

Listing 4: is vowel

```
1  
2 let is_vowel c =  
3 match c with  
4 'a'|'o'|'u'|'e'|'i' -> true  
5 | _ -> false  
6 ;;
```

Listing 5: isnil

```
1  
2 let isnil list =  
3     match list with  
4         | [] -> true  
5         | _ -> false  
6 ;;
```

Listing 6: length of a list

```

1
2 let rec length list =
3   match list with
4     [] -> 0
5     | h::t -> 1 + length t
6   ;;

```

Listing 7: reverse a list

```

1
2 let rec rev list =
3   match list with
4     [] -> []
5     | h::t -> rev t @ [h]
6   ;;

```

Listing 8: sum of a list of integers

```

1 let rec sum list =
2   match list with
3     [] -> 0
4     | h::t -> h + sum t
5   ;;

```

Listing 9: Append a list to another list

```

1 let rec append a b =
2   match a with
3     [] -> b
4     | h::t -> h::append t b
5   ;;

```

Listing 10: A list of integer in a given range

```

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

```

Listing 11: range 5 10

```

1 let r = range 5 10;;

```

Listing 12: first integer of the list

```

1 let first l =
2   match l with
3     | [] -> 0
4     | h::_ -> h;;

```

Listing 13: last integer of the list

```

1 let rec last l =
2   match l with
3     | [] -> 0
4     | [x]->x
5     | h::t->last t
6   ;;

```

Listing 14: factorial

```

1 let rec fact n =
2   if n = 0 then 1
3   else n * fact (n-1);;

```

Listing 15: concat a list

```

1 let rec concat l =
2   match l with
3     | [] -> ""
4     | h::t->h ^ concat t;;

```

Listing 16: map

```

1 let rec map f l =
2   match l with
3     | [] -> []
4     | h::t-> f h::(map f t)
5   ;;

```

Listing 17: fold

```

1 let rec fold (f,a,l) =
2   match l with
3     | []->a | (h::t)->fold (f,f(a,h),t);;
4
5 let next (a,-)=a+1;;
6 fold (next, 0, [1;2;3;4;6]);;

```

Listing 18: reverse a list using fold

```

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

```

Listing 19: sum of a list

```

1 let sum list=
2   fold ((fun(a,x)->a+x),0, list)

```

Listing 20: sum of a list

```

1 let sum list=
2   let add (a,x)=a+x in
3     fold (add,0, list)
4   ;;

```

Listing 21: merge 2 lists

```

1 let rec merge l1 l2 =
2   match l1 with
3     [] -> l2
4     | a::t -> h::merge l2 t;;

```

Listing 22: insert an item to a sorted list

```

1 let rec insert x l =
2   match l with
3     [] -> [x]
4     | h::t -> if x < h then x::h::t
5               else h::insert x t;;

```

Listing 23: insertion sort

```

1 let rec sort l =
2   match l with
3     | [] -> [];
4     | h::t -> print_int h; insert h (sort t)
5   ;;

```

1.1.1 Number to Word

Listing 24: Number to Word Conversion

```

1 (*
2   This program converts a number to the english word
3   15 => fifteen
4   123 => one hundred twenty three
5 *)
6
7 let get_ones x =
8   match x with
9     | 0 -> ""
10    | 1 -> "one"
11    | 2 -> "two"
12    | 3 -> "three"
13    | 4 -> "four"
14    | 5 -> "five"
15    | 6 -> "six"
16    | 7 -> "seven"
17    | 8 -> "eight"
18    | 9 -> "nine"
19    | 10 -> "ten"
20    | 11 -> "eleven"
21    | 12 -> "twelve"
22    | 13 -> "thirteen"
23    | 14 -> "fourteen"
24    | 15 -> "fifteen"

```

```
25         |16 ->"sixteen"
26         |17 ->"seventeen"
27         |18 ->"eighteen"
28         |19 ->"nineteen"
29         |_->"
30     ;;
31
32     let get_tens x =
33         match x with
34         |2 ->"twenty"
35         |3->"thirty"
36         |4->"forty"
37         |5->"fifty"
38         |6->"sixty"
39         |7->"seventy"
40         |8->"eighty"
41         |9->"ninety"
42         |_->"
43     ;;
44
45
46     let rec convert num =
47         let aux (d, str)=
48             let t1 = num / d in
49             let t2 = num mod d in
50             (convert t1) ^ str ^ (convert t2) in
51         if num >= 1000000000 then
52             aux (1000000000, "_billion_")
53         else if num >= 1000000 then
54             aux (1000000, "_million_")
55         else if num >= 1000 then
56             aux (1000, "_thousand_")
57         else if num >= 100 then
58             aux (100, "_hundred_")
59         else if num >= 20 then
60             let t1 = num / 10 in
61             let t2 = num mod 10 in
62             (get_tens t1) ^ "_" ^ (convert t2)
63         else
64             get_ones num
65     ;;
66
67
68     let n = 30;;
69     print_int n;;
70     print_newline ();;
71     print_string (convert n);;
72     print_newline ();;
```

References

[OCaml from the very beginning] JOHN WHITTINGTON *Coherent Press*