プログラミング B 課題 1

諏訪 凌太

2018 年 5 月 14 日

1 フィボナッチ数を計算する関数 fib

1.1 ソースプログラム

Listing 1: fib.ml

```
(* O(n) Algorithm *)
let rec auxfunc a b c =
  if c <= 2 then a
  else auxfunc (a+b) a (c-1);;

let fib n =
  auxfunc 1 1 n;;

for i = 1 to 30 do
  Printf.printf "%d\n" (fib i)
done;;
```

1.2 実行結果

```
   % ocaml fib.ml
   1
   1
   2
   3
   5
   8
   13
   21
   34
   55
   89
   144
   233
   377
   610
   987
   1597
   2584
   4181
   6765
   10946
   17711
```

```
    28657
    46368
    75025
    121393
    196418
    317811
    514229
    832040
```

2 2 つのリストのどちらにも現れる要素を集めたリストを返す関数 intersection

2.1 ソースプログラム

Listing 2: intersection.ml

```
open Printf

let rec member a ls =
  match ls with
  | [] -> false
  | h::t -> (a=h) || member a t;;

let rec intersection a b =
  match a with
  | [] -> []
  | h::t ->
    if member h b
    then h::intersection t b
    else intersection t b;;

let ans = intersection ([1;3;6;10;4])([10;3]);;
let () = List.iter (printf "%d ") ans;;
```

2.2 実行結果

```
    % ocaml intersection.ml
    3 10
```

3 最初のリストには現れるが 2 つ目のリストには現れない要素のリストを返す関数 difference

3.1 ソースプログラム

Listing 3: difference.ml

```
open Printf

let rec member a ls =
  match ls with
  | [] -> false
```

```
  | h::t -> (a=h) || member a t;;

let rec difference a b =
  match a with
  | [] -> []
  | h::t ->
    if member h b
    then difference t b
    else h::difference t b;;

let ans = difference ([1;3;6;10;4])([1;3]);;
let () = List.iter (printf "%d ") ans;;
```

3.2 実行結果

```
    % ocaml difference.ml
    6 10 4
```

4 1 つのリストの重複する要素を取り除いたリストを返す関数 unduplicate

4.1 ソースプログラム

Listing 4: unduplicate.ml

```
open Printf

let rec member a ls =
  match ls with
  | [] -> false
  | h::t -> (a=h) || member a t;;

let rec unduplicate a =
  match a with
  | [] -> []
  | h::t ->
    if member h t
    then unduplicate t
    else h::unduplicate t;;

let ans = unduplicate([1;4;6;1;6;6;7;7;7;7;7;8;9;0]);;
let () = List.iter (printf "%d ") ans;;
```

4.2 実行結果

```
    % ocaml unduplicate.ml
    4 1 6 7 8 9 0
```

5 exp(n) を計算するプログラム

5.1 ソースプログラム

Listing 5: exponential.ml

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

let sum = List.fold_left ( +. ) 0.0;;
let product = List.fold_left ( * ) 1;;

let fact n = product (range 1 n);;


let exp n =
  let rec sum_exp a n =
    if a > n then [1.0]
    else 1.0 /. (float_of_int (fact a)) :: sum_exp (a+1) n
  in
  sum (sum_exp 1 n);;

let ans = exp 20;;
print_float ans;;
```

5.2 実行結果

```
    % ocaml exponential.ml
    2.71828182846
```

6 与えられた 0 個以上の 1 桁の整数の和を印刷する C プログラム

6.1 ソースプログラム

```
/**
 * ※使用禁止
 * 1. 変数への破壊的代入 (=, +=, ++, etc)
 * 2. for, while, do while, goto, longjmp
 * 3. 以外の関数定義 main
 *
 * @brief ./a.out 3 1 4 1 5 のように与えられた整数の和を計算します 9
 * @author Ryota Suwa
 * Licenced in CC0
 */

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
  if (argv[argc] == NULL) {
    if (argc == 1) {
      fprintf(stderr, "Usage: %s num1 num2 ... numN\n", argv[0]);
      exit(EXIT_FAILURE);
    }
    printf("%d\n", atoi(argv[argc - 1]) + main(argc - 1, argv));
  }
  if (argc > 1) {
    return atoi(argv[argc - 1]) + main(argc - 1, argv);
  }
  return 0;
```

```
27 }
```

6.2 実行結果

```
1     % ./a.out 3 3 1 4 3 5 1 3 9
2     32
```

7 OCaml プログラミングの感想

最初は戸惑ったが，慣れてくると手続き型言語よりも直感的に書けるようになった．fib n を求めるプログラムでは，行列を用いることにより O(logn) で計算することができるが，今回は，公式を用いて O(1) で求めるという強引なプログラムを書いたので，以下に示す．

7.1 ソースプログラム

Listing 6: fib2.ml

```
1  (* O(1) Algorithm *)
2
3  let rec pow (x, y) =
4    if y = 0 then 1.0
5    else
6      let z = pow (x, y / 2) in
7      let zz = z *. z in
8      if y mod 2 = 0 then zz else x *. zz
9
10 let round n = floor (n +. 0.5)
11
12 let fib n =
13   int_of_float (round (((pow((1.0 +. sqrt 5.0) /. 2.0, n) -. pow((1.0 -. sqrt 5.0) /. 2.0,
         n)) /. sqrt(5.0))));;
14
15 for i = 1 to 40 do
16   Printf.printf "%d\n" (fib i)
17 done;;
```

7.2 実行結果

```
1     % ocaml fib2.ml
2     1
3     1
4     2
5     3
6     5
7     8
8     13
9     21
10    34
11    55
12    89
13    144
14    233
```

```
15      377
16      610
17      987
18      1597
19      2584
20      4181
21      6765
22      10946
23      17711
24      28657
25      46368
26      75025
27      121393
28      196418
29      317811
30      514229
31      832040
32      1346269
33      2178309
34      3524578
35      5702887
36      9227465
37      14930352
38      24157817
39      39088169
40      63245986
41      102334155
```

今回用いたソースコードは GitHub に公開した．リンクを以下に示す．

`https://github.com/Enantiomer/OCaml_ProB`