

宣言型プログラム論 課題6

200911434 青木大祐

平成 24 年 10 月 23 日

6-1

```
1 module type QUEUE =
2 sig
3   type 'a t
4   val empty: 'a t
5   val enq : 'a t -> 'a -> 'a t
6   val null : 'a t -> bool
7   val deq : 'a t -> 'a * 'a t
8 end;;
9
10 module Queue3:QUEUE =
11 struct
12   type 'a t = 'a list * 'a list
13   let empty = ([], [])
14   let norm (l1, l2) =
15     match l1 with
16     | [] -> (List.rev l2, [])
17     | _ -> (l1, l2)
18   let enq (l1, l2) x = norm (l1, x::l2)
19   let null q =
20     match q with
21     | ([],[]) -> true
22     | _ -> false
23   let deq (l1,l2) =
24     match l1 with
25     | x::l1' -> (x, norm (l1', l2))
26     | _ -> raise (Failure "deq")
27 end
28
29 (* 6.1.1 *)
30 let enqlist q list =
31   List.fold_left (fun queue x -> Queue3.enq queue x) q list
32 ;;
33
34 (* 6.1.2 *)
35 let print_int_queue q =
36   let tmp = ref q in
37   try
38     while true do
39       let x,queue = Queue3.deq !tmp in
40       tmp := queue;
41       print_int x;
42       print_string "\n"
43     done
44   with Failure "deq" -> ()
45 ;;
46
47 let q = enqlist Queue3.empty [1;2;3];;
48 let x,q = Queue3.deq q;;
49 let x,q = Queue3.deq q;;
50 let x,q = Queue3.deq q;;
51
52 let q = enqlist Queue3.empty [1;2;3];;
53 print_int_queue q;;
```

実行結果

```
1 # val enqlist : 'a Queue3.t -> 'a list -> 'a Queue3.t = <fun>
2 # val print_int_queue : int Queue3.t -> unit = <fun>
3 # val q : int Queue3.t = <abstr>
4 # val x : int = 1
5 # val q : int Queue3.t = <abstr>
6 # val x : int = 2
7 # val q : int Queue3.t = <abstr>
8 # val x : int = 3
9 # val q : int Queue3.t = <abstr>
10 # val q : int Queue3.t = <abstr>
11 # 1
12 2
13 3
14 - : unit = ()
```

6-2

```
1 (* question 6.2 *)
2 module type QUEUE =
3 sig
```

```

4   type 'a t
5   val empty: 'a t
6   val enq : 'a t -> 'a -> 'a t
7   val null : 'a t -> bool
8   val deq : 'a t -> 'a * 'a t
9 end;;
10
11 module Queue3:QUEUE =
12 struct
13   type 'a t = 'a list * 'a list
14   let empty = ([], [])
15   let norm (l1, l2) =
16     match l1 with
17     [] -> (List.rev l2, [])
18     | _ -> (l1, l2)
19   let enq (l1, l2) x = norm (l1, x::l2)
20   let null q =
21     match q with
22     ([],[]) -> true
23     | _ -> false
24   let deq (l1,l2) =
25     match l1 with
26     x::l1' -> (x, norm (l1', l2))
27     | _ -> raise (Failure "deq")
28 end
29
30 let enqlist q list =
31   List.fold_left (fun queue x -> Queue3.enq queue x) q list
32 ;;
33
34 let print_int_queue q =
35   let tmp = ref q in
36   try
37     while true do
38       let x,queue = Queue3.deq !tmp in
39       tmp := queue;
40       print_int x;
41       print_string "\n"
42     done
43   with Failure "deq" -> ()
44 ;;
45
46 let graph1 = [("a", "b"); ("a", "c"); ("a", "d");
47              ("b", "e"); ("c", "f"); ("d", "e");
48              ("e", "f"); ("e", "g"); ("f", "d")]
49
50 ;;
51
52 let nexts v xs =
53   List.fold_right (fun x l ->
54     let (a, b) = x in
55     if a = v then b::l else l) xs []
56 ;;
57
58 (* 6.2.1 *)
59 nexts "a" graph1;;
60
61 let mem x ys = List.exists (fun y -> y = x) ys;;
62
63 let search graph v =
64   let rec bfs q vs = (* bfs: breadth first search *)
65     if Queue3.null q then vs
66     else
67       let (v, q') = Queue3.deq q in
68       if mem v vs then bfs q' vs
69       else
70         let vs = v::vs in
71         let q = enqlist q' (nexts v graph) in
72         bfs q vs
73   in
74   bfs (Queue3.enq Queue3.empty v) []
75 ;;
76
77 (* 6.6.2 *)
78 search graph1 "a";;

```

実行結果

```

1 # val enqlist : 'a Queue3.t -> 'a list -> 'a Queue3.t = <fun>
2 # val print_int_queue : int Queue3.t -> unit = <fun>
3 # val graph1 : (string * string) list =
4 # [("a", "b"); ("a", "c"); ("a", "d"); ("b", "e"); ("c", "f"); ("d", "e");
5 #   ("e", "f"); ("e", "g"); ("f", "d")]
6 # val nexts : 'a -> ('a * 'b) list -> 'b list = <fun>
7 # - : string list = ["b"; "c"; "d"]

```

```
8 | # val mem : 'a -> 'a list -> bool = <fun>
9 | # val search : ('a * 'a) list -> 'a -> 'a list = <fun>
10| # - : string list = ["g"; "f"; "e"; "d"; "c"; "b"; "a"]
```