

Objective Caml version 3.09.0

```

(* ----- *)
(* Endliche Mengen werden als Listen ohne Wiederholung implementiert. *)
(* ----- *)

# type 'a set = 'a list;;

type 'a set = 'a list

(* ----- *)
(* add x s liefert die Vereinigung von s und {x}, wenn x nicht in s liegt. *)
(* ----- *)

# let add (x: 'a) (s: 'a set): 'a set = x :: s;;

val add : 'a -> 'a set -> 'a set = <fun>

(* ----- *)
(* disjoint_union s1 s2 liefert die Vereinigung von s1 und s2, wenn s1 und s2 disjunkt sind. *)
(* ----- *)

# let disjoint_union (s1: 'a set) (s2:'a set): 'a set = s1 @ s2;;

val disjoint_union : 'a set -> 'a set -> 'a set = <fun>

(* ----- *)
(* big_disjoint_union liefert die Vereinigung einer Liste von disjunkten Mengen. *)
(* ----- *)

# let big_disjoint_union (l: 'a set list): 'a set = List.flatten l;;

val big_disjoint_union : 'a set list -> 'a set = <fun>

(* ----- *)
(* powerset(s) liefert die Potenzmenge der Menge s. *)
(* ----- *)

# let rec powerset (s: 'a set): 'a set set =
  match s with
  [] -> [[]]
  | (x :: s') -> let p = powerset s' in disjoint_union p (List.map (add x) p);;

val powerset : 'a set -> 'a set set = <fun>

# powerset [];;
- : 'a set set = [[]]

# powerset [1];;
- : int set set = [[]; [1]]

# powerset [1;2];;
- : int set set = [[]; [2]; [1]; [1; 2]]

# powerset [1;2;3];;
- : int set set = [[]; [3]; [2]; [2; 3]; [1]; [1; 3]; [1; 2]; [1; 2; 3]]

# powerset [1;2;3;4];;
- : int set set =
[[]; [4]; [3]; [3; 4]; [2]; [2; 4]; [2; 3]; [2; 3; 4]; [1]; [1; 4]; [1; 3];
 [1; 3; 4]; [1; 2]; [1; 2; 4]; [1; 2; 3]; [1; 2; 3; 4]]

(* ----- *)
(* difference s1 s2 liefert die Mengendifferenz s1 \ s2. *)
(* ----- *)

# let rec difference (s1: 'a set) (s2: 'a set): 'a set =

```

```

match s1 with
  []      -> []
  | x :: s -> if List.mem x s2 then difference s s2 else x :: difference s s2;;

val difference : 'a set -> 'a set -> 'a set = <fun>

# difference [1;2;3;4] [];;
- : int set = [1; 2; 3; 4]

# difference [1;2;3;4] [2];;
- : int set = [1; 3; 4]

# difference [1;2;3;4] [2;4];;
- : int set = [1; 3]

# difference [1;2;3;4] [1;2;3];;
- : int set = [4]

(* ----- *)
(* partitions s liefert die Menge Part(s) aller Partitionen von s nach der in Aufgabe 2 ange- *)
(* gebenen "Rekursionsformel". *)
(* ----- *)

# let rec partitions (s: 'a set) =
  match s with
  []      -> [[]]
  | x :: s' ->
    let p = powerset s' in
    big_disjoint_union (List.map (fun b -> List.map (add (add x b)) (partitions (difference s' b))) p);;

val partitions : 'a set -> 'a set set set = <fun>

(* ----- *)
(* Die restlichen Funktionen sorgen für eine vernünftige Ausgabe der Mengen. *)
(* ----- *)

# let string_of_int_set (s: int set): string =
  "{" ^ String.concat ", " (List.map string_of_int s) ^ "}";;

val string_of_int_set : int set -> string = <fun>

# let string_of_partition (p: int set set): string =
  "{" ^ String.concat ", " (List.map string_of_int_set p) ^ "}";;

val string_of_partition : int set set -> string = <fun>

# let print_partitions (s: int set) =
  let ps = partitions s in
  print_string ("\nDie Menge " ^ string_of_int_set s ^ " besitzt " ^ string_of_int (List.length ps) ^ " Partitionen:\n\n");
  List.iter (fun p -> print_string (string_of_partition p ^ ",\n")) ps;
  print_string "\n";;

val print_partitions : int set -> unit = <fun>

# print_partitions [];;

Die Menge {} besitzt 1 Partitionen:

{},
- : unit = ()

# print_partitions [1];;

```

Die Menge {1} besitzt 1 Partitionen:

```
{1},
- : unit = ()
# print_partitions [1;2];;
```

Die Menge {1, 2} besitzt 2 Partitionen:

```
{1}, {2},
{1, 2},
- : unit = ()
# print_partitions [1;2;3];;
```

Die Menge {1, 2, 3} besitzt 5 Partitionen:

```
{1}, {2}, {3},
{1}, {2, 3},
{1, 3}, {2},
{1, 2}, {3},
{1, 2, 3},
- : unit = ()
# print_partitions [1;2;3;4];;
```

Die Menge {1, 2, 3, 4} besitzt 15 Partitionen:

```
{1}, {2}, {3}, {4},
{1}, {2}, {3, 4},
{1}, {2, 4}, {3},
{1}, {2, 3}, {4},
{1}, {2, 3, 4},
{1, 4}, {2}, {3},
{1, 4}, {2, 3},
{1, 3}, {2}, {4},
{1, 3}, {2, 4},
{1, 3, 4}, {2},
{1, 2}, {3}, {4},
{1, 2}, {3, 4},
{1, 2, 4}, {3},
{1, 2, 3}, {4},
{1, 2, 3, 4},
- : unit = ()
# print_partitions [1;2;3;4;5];;
```

Die Menge {1, 2, 3, 4, 5} besitzt 52 Partitionen:

```
{1}, {2}, {3}, {4}, {5},
{1}, {2}, {3}, {4, 5},
{1}, {2}, {3, 5}, {4},
{1}, {2}, {3, 4}, {5},
{1}, {2}, {3, 4, 5},
{1}, {2, 5}, {3}, {4},
{1}, {2, 5}, {3, 4},
{1}, {2, 4}, {3}, {5},
{1}, {2, 4}, {3, 5},
{1}, {2, 4, 5}, {3},
{1}, {2, 3}, {4}, {5},
{1}, {2, 3}, {4, 5},
{1}, {2, 3, 5}, {4},
{1}, {2, 3, 4}, {5},
{1}, {2, 3, 4, 5},
```

```

{1, 5}, {2}, {3}, {4}},
{1, 5}, {2}, {3, 4}},
{1, 5}, {2, 4}, {3}},
{1, 5}, {2, 3}, {4}},
{1, 5}, {2, 3, 4}},
{1, 4}, {2}, {3}, {5}},
{1, 4}, {2}, {3, 5}},
{1, 4}, {2, 5}, {3}},
{1, 4}, {2, 3}, {5}},
{1, 4}, {2, 3, 5}},
{1, 4, 5}, {2}, {3}},
{1, 4, 5}, {2, 3}},
{1, 3}, {2}, {4}, {5}},
{1, 3}, {2}, {4, 5}},
{1, 3}, {2, 5}, {4}},
{1, 3}, {2, 4}, {5}},
{1, 3}, {2, 4, 5}},
{1, 3, 5}, {2}, {4}},
{1, 3, 5}, {2, 4}},
{1, 3, 4}, {2}, {5}},
{1, 3, 4}, {2, 5}},
{1, 3, 4, 5}, {2}},
{1, 2}, {3}, {4}, {5}},
{1, 2}, {3}, {4, 5}},
{1, 2}, {3, 5}, {4}},
{1, 2}, {3, 4}, {5}},
{1, 2}, {3, 4, 5}},
{1, 2, 5}, {3}, {4}},
{1, 2, 5}, {3, 4}},
{1, 2, 4}, {3}, {5}},
{1, 2, 4}, {3, 5}},
{1, 2, 4, 5}, {3}},
{1, 2, 3}, {4}, {5}},
{1, 2, 3}, {4, 5}},
{1, 2, 3, 5}, {4}},
{1, 2, 3, 4}, {5}},
{1, 2, 3, 4, 5}},

```

```

- : unit = ()
int_partitions [1;2;3;4;5;6];;

(* print_partitions [1;2;3;4;5;6;7];; *)
(* print_partitions [1;2;3;4;5;6;7;8];; *)
(* print_partitions [1;2;3;4;5;6;7;8;9];; *)
(* print_partitions [1;2;3;4;5;6;7;8;9;10];; *)

# print_partitions [1;2;3;4;5;6];;

```

Die Menge {1, 2, 3, 4, 5, 6} besitzt 203 Partitionen:

```

{1}, {2}, {3}, {4}, {5}, {6}},
{1}, {2}, {3}, {4}, {5, 6}},
{1}, {2}, {3}, {4, 6}, {5}},
{1}, {2}, {3}, {4, 5}, {6}},
{1}, {2}, {3}, {4, 5, 6}},
{1}, {2}, {3, 6}, {4}, {5}},
{1}, {2}, {3, 6}, {4, 5}},
{1}, {2}, {3, 5}, {4}, {6}},
{1}, {2}, {3, 5}, {4, 6}},
{1}, {2}, {3, 5, 6}, {4}},
{1}, {2}, {3, 4}, {5}, {6}},
{1}, {2}, {3, 4}, {5, 6}},
{1}, {2}, {3, 4, 6}, {5}},
{1}, {2}, {3, 4, 5}, {6}},
{1}, {2}, {3, 4, 5, 6}},
{1}, {2, 6}, {3}, {4}, {5}},
{1}, {2, 6}, {3}, {4, 5}},
{1}, {2, 6}, {3, 5}, {4}},
{1}, {2, 6}, {3, 4}, {5}},
{1}, {2, 6}, {3, 4, 5}},
{1}, {2, 5}, {3}, {4}, {6}},

```

{1}, {2, 5}, {3}, {4, 6},
{1}, {2, 5}, {3, 6}, {4},
{1}, {2, 5}, {3, 4}, {6},
{1}, {2, 5}, {3, 4, 6},
{1}, {2, 5, 6}, {3}, {4},
{1}, {2, 5, 6}, {3, 4},
{1}, {2, 4}, {3}, {5}, {6},
{1}, {2, 4}, {3}, {5, 6},
{1}, {2, 4}, {3, 6}, {5},
{1}, {2, 4}, {3, 5}, {6},
{1}, {2, 4}, {3, 5, 6},
{1}, {2, 4, 6}, {3}, {5},
{1}, {2, 4, 6}, {3, 5},
{1}, {2, 4, 5}, {3}, {6},
{1}, {2, 4, 5}, {3, 6},
{1}, {2, 4, 5, 6}, {3},
{1}, {2, 3}, {4}, {5}, {6},
{1}, {2, 3}, {4}, {5, 6},
{1}, {2, 3}, {4, 6}, {5},
{1}, {2, 3}, {4, 5}, {6},
{1}, {2, 3}, {4, 5, 6},
{1}, {2, 3, 6}, {4}, {5},
{1}, {2, 3, 6}, {4, 5},
{1}, {2, 3, 5}, {4}, {6},
{1}, {2, 3, 5}, {4, 6},
{1}, {2, 3, 5, 6}, {4},
{1}, {2, 3, 4}, {5}, {6},
{1}, {2, 3, 4}, {5, 6},
{1}, {2, 3, 4, 6}, {5},
{1}, {2, 3, 4, 5}, {6},
{1}, {2, 3, 4, 5, 6},
{1, 6}, {2}, {3}, {4}, {5},
{1, 6}, {2}, {3}, {4, 5},
{1, 6}, {2}, {3, 5}, {4},
{1, 6}, {2}, {3, 4}, {5},
{1, 6}, {2}, {3, 4, 5},
{1, 6}, {2, 5}, {3}, {4},
{1, 6}, {2, 5}, {3, 4},
{1, 6}, {2, 4}, {3}, {5},
{1, 6}, {2, 4}, {3, 5},
{1, 6}, {2, 4, 5}, {3},
{1, 6}, {2, 3}, {4}, {5},
{1, 6}, {2, 3}, {4, 5},
{1, 6}, {2, 3, 5}, {4},
{1, 6}, {2, 3, 4}, {5},
{1, 6}, {2, 3, 4, 5},
{1, 5}, {2}, {3}, {4}, {6},
{1, 5}, {2}, {3}, {4, 6},
{1, 5}, {2}, {3, 6}, {4},
{1, 5}, {2}, {3, 4}, {6},
{1, 5}, {2}, {3, 4, 6},
{1, 5}, {2, 6}, {3}, {4},
{1, 5}, {2, 6}, {3, 4},
{1, 5}, {2, 4}, {3}, {6},
{1, 5}, {2, 4}, {3, 6},
{1, 5}, {2, 4, 6}, {3},
{1, 5}, {2, 3}, {4}, {6},
{1, 5}, {2, 3}, {4, 6},
{1, 5}, {2, 3, 6}, {4},
{1, 5}, {2, 3, 4}, {6},
{1, 5}, {2, 3, 4, 6},
{1, 5, 6}, {2}, {3}, {4},
{1, 5, 6}, {2}, {3, 4},
{1, 5, 6}, {2, 4}, {3},
{1, 5, 6}, {2, 3}, {4},
{1, 5, 6}, {2, 3, 4},
{1, 4}, {2}, {3}, {5}, {6},
{1, 4}, {2}, {3}, {5, 6},
{1, 4}, {2}, {3, 6}, {5},
{1, 4}, {2}, {3, 5}, {6},
{1, 4}, {2}, {3, 5, 6},
{1, 4}, {2, 6}, {3}, {5},

{1, 4}, {2, 6}, {3, 5},
{1, 4}, {2, 5}, {3, 6},
{1, 4}, {2, 5}, {3, 6},
{1, 4}, {2, 5, 6}, {3},
{1, 4}, {2, 3}, {5}, {6},
{1, 4}, {2, 3}, {5, 6},
{1, 4}, {2, 3, 6}, {5},
{1, 4}, {2, 3, 5}, {6},
{1, 4}, {2, 3, 5, 6},
{1, 4, 6}, {2}, {3}, {5},
{1, 4, 6}, {2}, {3, 5},
{1, 4, 6}, {2, 5}, {3},
{1, 4, 6}, {2, 3}, {5},
{1, 4, 6}, {2, 3, 5},
{1, 4, 5}, {2}, {3}, {6},
{1, 4, 5}, {2}, {3, 6},
{1, 4, 5}, {2, 6}, {3},
{1, 4, 5}, {2, 3}, {6},
{1, 4, 5}, {2, 3, 6},
{1, 4, 5, 6}, {2}, {3},
{1, 4, 5, 6}, {2, 3},
{1, 3}, {2}, {4}, {5}, {6},
{1, 3}, {2}, {4}, {5, 6},
{1, 3}, {2}, {4, 6}, {5},
{1, 3}, {2}, {4, 5}, {6},
{1, 3}, {2}, {4, 5, 6},
{1, 3}, {2, 6}, {4}, {5},
{1, 3}, {2, 6}, {4, 5},
{1, 3}, {2, 5}, {4}, {6},
{1, 3}, {2, 5, 6}, {4},
{1, 3}, {2, 4}, {5}, {6},
{1, 3}, {2, 4}, {5, 6},
{1, 3}, {2, 4, 6}, {5},
{1, 3}, {2, 4, 5}, {6},
{1, 3}, {2, 4, 5, 6},
{1, 3, 6}, {2}, {4}, {5},
{1, 3, 6}, {2}, {4, 5},
{1, 3, 6}, {2, 5}, {4},
{1, 3, 6}, {2, 4}, {5},
{1, 3, 6}, {2, 4, 5},
{1, 3, 5}, {2}, {4}, {6},
{1, 3, 5}, {2}, {4, 6},
{1, 3, 5}, {2, 6}, {4},
{1, 3, 5}, {2, 4}, {6},
{1, 3, 5}, {2, 4, 6},
{1, 3, 5, 6}, {2}, {4},
{1, 3, 5, 6}, {2, 4},
{1, 3, 4}, {2}, {5}, {6},
{1, 3, 4}, {2}, {5, 6},
{1, 3, 4}, {2, 6}, {5},
{1, 3, 4}, {2, 5}, {6},
{1, 3, 4}, {2, 5, 6},
{1, 3, 4, 6}, {2}, {5},
{1, 3, 4, 6}, {2, 5},
{1, 3, 4, 5}, {2}, {6},
{1, 3, 4, 5}, {2, 6},
{1, 3, 4, 5, 6}, {2},
{1, 2}, {3}, {4}, {5}, {6},
{1, 2}, {3}, {4}, {5, 6},
{1, 2}, {3}, {4, 6}, {5},
{1, 2}, {3}, {4, 5}, {6},
{1, 2}, {3}, {4, 5, 6},
{1, 2}, {3, 6}, {4}, {5},
{1, 2}, {3, 6}, {4, 5},
{1, 2}, {3, 5}, {4}, {6},
{1, 2}, {3, 5}, {4, 6},
{1, 2}, {3, 5, 6}, {4},
{1, 2}, {3, 4}, {5}, {6},
{1, 2}, {3, 4}, {5, 6},
{1, 2}, {3, 4, 6}, {5},
{1, 2}, {3, 4, 5}, {6}

```

{{1, 2}, {3, 4, 5, 6}},
{{1, 2, 6}, {3}, {4}, {5}},
{{1, 2, 6}, {3}, {4, 5}},
{{1, 2, 6}, {3, 5}, {4}},
{{1, 2, 6}, {3, 4}, {5}},
{{1, 2, 6}, {3, 4, 5}},
{{1, 2, 5}, {3}, {4}, {6}},
{{1, 2, 5}, {3}, {4, 6}},
{{1, 2, 5}, {3, 6}, {4}},
{{1, 2, 5}, {3, 4}, {6}},
{{1, 2, 5}, {3, 4, 6}},
{{1, 2, 5, 6}, {3}, {4}},
{{1, 2, 5, 6}, {3, 4}},
{{1, 2, 4}, {3}, {5}, {6}},
{{1, 2, 4}, {3}, {5, 6}},
{{1, 2, 4}, {3, 6}, {5}},
{{1, 2, 4}, {3, 5}, {6}},
{{1, 2, 4}, {3, 5, 6}},
{{1, 2, 4, 6}, {3}, {5}},
{{1, 2, 4, 6}, {3, 5}},
{{1, 2, 4, 5}, {3}, {6}},
{{1, 2, 4, 5}, {3, 6}},
{{1, 2, 4, 5, 6}, {3}},
{{1, 2, 3}, {4}, {5}, {6}},
{{1, 2, 3}, {4}, {5, 6}},
{{1, 2, 3}, {4, 6}, {5}},
{{1, 2, 3}, {4, 5}, {6}},
{{1, 2, 3}, {4, 5, 6}},
{{1, 2, 3, 6}, {4}, {5}},
{{1, 2, 3, 6}, {4, 5}},
{{1, 2, 3, 5}, {4}, {6}},
{{1, 2, 3, 5}, {4, 6}},
{{1, 2, 3, 5, 6}, {4}},
{{1, 2, 3, 4}, {5}, {6}},
{{1, 2, 3, 4}, {5, 6}},
{{1, 2, 3, 4, 6}, {5}},
{{1, 2, 3, 4, 5}, {6}},
{{1, 2, 3, 4, 5, 6}},

```

```
- : unit = ()
```

```
#
```