Zadanie B2: SML - 2-3 drzewa (remove)

2-3 drzewa to drzewa poszukiwań binarnych w których każdy węzeł jest albo:

- Liściem (pustym)
- Węzłem z jednym kluczem i parą synów
- Węzłem z parą kluczy i trójką synów

W SML-u można to reprezentować przez następujący typ danych:

```
datatype 'a node = Two of 'a node * 'a * 'a node |
                 Three of 'a node * 'a * 'a node * 'a * 'a node |
                 Empty;
```

Liście 2-3 drzewa znajdują się na jednym poziomie. W SML-u warunek ten można sprawdzić poniższą funkcją check:

```
(* compute height *)

local
  fun height' (Empty, h) = h |
      height' (Two(t1, _, _), h) = height' (t1, h+1) |
      height' (Three(t1, _, _, _, _), h) = height' (t1, h+1);
in
  fun height t = height' (t, 0);
end

(* check if height is ok *)

local
  fun check' (Empty, 0) = true |
      check' (Empty, h) = false |
      check' (Two(t1, _, t2), h) = check' (t1, h-1) andalso check' (t2, h-1) |
      check' (Three(t1, _, t2, _, t3), h) = check' (t1, h-1) andalso check' (t2, h-1) andalso check'
in
  fun check T = check' (T, height T)
end
```

2-3 drzewo zachowuje porządek inorder elementów. Dzięki temu poniższa funkcja sprawdzająca obecność wartości w drzewie działa w czasie $O(\log n)$:

```
(* search *)

fun search cmp (x, Empty) = false |
  search cmp (x, Two (left, y, right)) =
    let val c = cmp (x, y);
    in
      if (c < 0) then
        search cmp (x, left)
      else if (c > 0) then
```

```
      search cmp (x, right)
    else true
  end |
search cmp (x, Three (left, y, middle, z, right)) =
  let val c1 = cmp (x, y);
  in
    if (c1 < 0) then
      search cmp (x, left)
    else if (c1 > 0) then
      let val c2 = cmp (x, z)
      in
        if (c2 < 0) then
          search cmp (x, middle)
        else if (c2 > 0) then
          search cmp (x, right)
        else true
      end
    else true
  end;
```

Argument *cmp* funkcji search to funkcja porównująca elementy, zwracająca wartość < 0, $= 0$, > 0 w zależności od relacji między argumentami. Taka funkcja może zostać zdefiniowana np. tak:

```
fun mycmp (x : int, y) = x - y;
```

Napisz funkcję remove : ('a∗'a → int) → 'a * 'a node → bool * 'a node, która usuwa wartość z drzewa. Pierwszy argument funkcji to funkcja porównująca, drugi to para (x, T), gdzie x to wstawiany element, a T to drzewo, do którego wstawiamy. Wartość zwracana przez fukcję to para, której pierwszy element to wartość true/false mówiąca czy element x znajdował się w drzewie, a drugi element to drzewo T bez elementu x.

W pliku .sml powinna znaleźć się definicja:

```
fun remove cmp (x, T) = ...
```

Uwaga

Rozwiązanie powinno być czysto funkcyjne. Rozwiązania bazujące na referencjach lub tablicach będą odrzucane.

Przykład

```
- remove mycmp (5, Three (Two (Empty,2,Empty),3,Two (Empty,4,Empty),5,Two (Empty,6,Empty)));
val it = (true,Two (Two (Empty,2,Empty),3,Three (Empty,4,Empty,6,Empty))) : bool * int node
```