TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK

Lehrstuhl für Sprachen und Beschreibungsstrukturen WS 2015/16
Einführung in die Informatik 2 **Übungsblatt 14**
Prof. Dr. Helmut Seidl, Ralf Vogler, Stefan Schulze Frielinghaus

BONUS-Hausaufgabenblatt: Von den 14 Hausaufgabenblättern müssen 10 bestanden sein, um einen Bonus auf die Klausur angerechnet zu bekommen.

Aufgabe [5 Punkte] **OCaml: Nebenläufigkeit**

Implementieren Sie ein Modul `Parallel` mit folgender Signatur und laden Sie Ihre Abgabe als `parallel.ml` auf Moodle hoch.

```
val map : ('a -> 'b) -> 'a list -> 'b list
```

Dabei soll bei `map f xs` für jedes Element der Liste `xs` die Funktion `f` in einem eigenen Thread aufgerufen werden und die Ergebnisse mit `Event.select` wieder kombiniert werden. Die Ordnung der Liste muss dabei erhalten bleiben!

Lösungsvorschlag 14.1

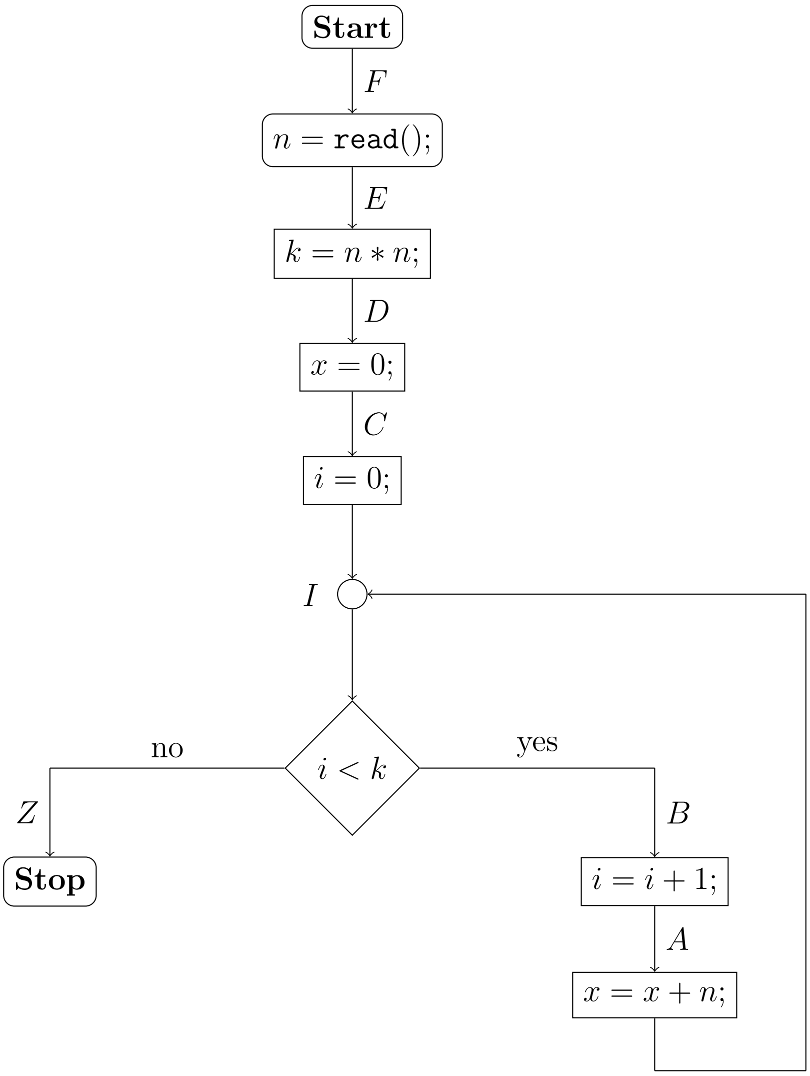
```
let comp2 f g x y = f (g x) (g y)
let compareBy f = comp2 compare f

let map : ('a -> 'b) -> 'a list -> 'b list = fun f xs ->
  let t f x =
    let c = Event.new_channel () in
    let _ = Thread.create (fun x -> Event.(sync (send c (f x)))) x in
    Event.receive c
  in
  let fs = List.mapi (fun i x -> Event.wrap (t f x) (fun y -> i,y)) xs in
  let ys = List.map (fun _ -> Event.select fs) fs in (* call select for each element...
  List.sort (compareBy fst) ys |> List.map snd

let test =
  let xs = [1;2;3] in
  let ys = map ((+) 1) xs in
  let string_of_list s xs = "[" ^ String.concat ", " (List.map s xs) ^ "]" in
  print_endline @@ string_of_list string_of_int ys
```

Aufgabe [5 Punkte] **Ein Leben ohne Zusicherungen? Unvorstellbar!**

Gegeben sei folgendes Kontroll-Fluß-Diagramm:



a) Zeigen Sie, dass am **Stop**-Knoten die Zusicherung $Z \equiv (x = n^3)$ stets erfüllt ist! Geben Sie dafür die Zusicherungen A bis F sowie I an und zeigen, dass diese lokal konsistent sind.

b) Zeigen Sie, dass das Programm stets terminiert, indem Sie an geeigneten Stellen im Programm eine Kenngröße r einführen und beweisen deren Gültigkeit.

Lösungsvorschlag 14.2

Teilaufgabe a) Setze $I :\equiv (i \leq k \land k = n^2 \land x = i * n)$. Dann ergibt sich:

$$\mathsf{WP}[\![x = x + n]\!](I) = (i \leq k \land k = n^2 \land x + n = i * n) =: A$$

$$\begin{array}{rcl}
\mathsf{WP}[\![i = i + 1]\!](A) & = & i + 1 \leq k \land k = n^2 \land x + n = (i + 1) * n \\
& = & i + 1 \leq k \land k = n^2 \land x + n = i * n + n \\
& = & i + 1 \leq k \land k = n^2 \land x = i * n =: B \\
& \Leftarrow & i < k \land I
\end{array}$$

$$\begin{array}{rcl}
\mathsf{WP}[\![i<k]\!](Z,B) & = & (i \geq k \wedge Z) \vee (i<k \wedge B) \\
 & = & (i \geq k \wedge x = n^3) \vee (i<k \wedge i+1 \leq k \wedge k = n^2 \wedge x = i*n) \\
 & = & (i \geq k \wedge x = n^3) \vee (i<k \wedge I) \\
 & \Leftarrow & (i \geq k \wedge k = n^2 \wedge x = k*n) \vee (i<k \wedge I) \\
 & \Leftarrow & (i \geq k \wedge i \leq k \wedge k = n^2 \wedge x = i*n) \vee (i<k \wedge I) \\
 & = & (i \geq k \wedge I) \vee (i<k \wedge I) \\
 & = & (i \geq k \vee i<k) \wedge I \\
 & = & I
\end{array}$$

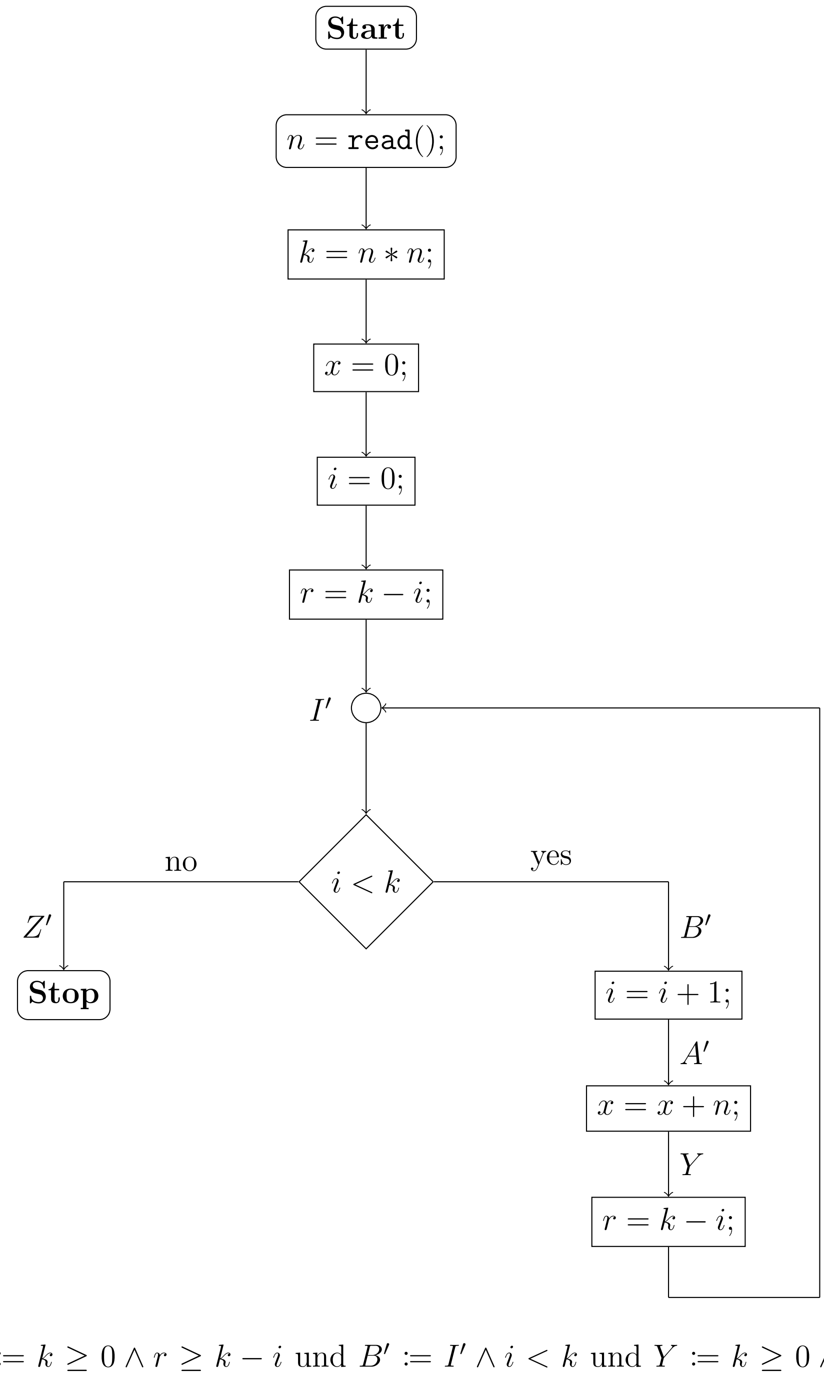
$\mathsf{WP}[\![i=0]\!](I) = (0 \leq k \wedge k = n^2 \wedge x = 0*n) = (0 \leq k \wedge k = n^2 \wedge x = 0) =: C$

$\mathsf{WP}[\![x=0]\!](C) = (0 \leq k \wedge k = n^2 \wedge 0 = 0) = (0 \leq k \wedge k = n^2) =: D$

$\mathsf{WP}[\![k=n*n]\!](D) = (0 \leq n*n \wedge n^2 = n^2) = \mathbf{true} =: E$

$\mathsf{WP}[\![n=\texttt{read}()]\!](E) = (\forall n.\mathbf{true}) = \mathbf{true} =: F$

Teilaufgabe b)



Wir setzen $I' := k \geq 0 \wedge r \geq k - i$ und $B' := I' \wedge i < k$ und $Y := k \geq 0 \wedge r > k - i$ und

$Z' :=$ **true**.

Dann gilt $B' \implies r \geq 0$ und wir folgern

$\mathsf{WP}[\![r = k - i]\!](I') \equiv k \geq 0 \land k - i \geq k - i \equiv k \geq 0 \impliedby Y$

$\mathsf{WP}[\![x = x + n]\!](Y) \equiv Y =: A'$

$\mathsf{WP}[\![i = i + 1]\!](A') \equiv k \geq 0 \land r > k - (i + 1) \impliedby B'$

$\mathsf{WP}[\![i < k]\!](Z', B') \equiv (i \geq k) \lor (i < k \land B') \impliedby B'$

$\mathsf{WP}[\![n = \texttt{read}(); k = n * n; x = 0; i = 0; r = k - i]\!](I') \equiv \textbf{true}$

D.h. wir haben gezeigt, dass beim Betreten der Schleife $r \geq 0$ gilt und bei jedem Schleifendurchlauf wird r kleiner. Somit muss das Programm terminieren.