Semantyka i weryfikacja programów 2023/24.
Egzamin 2/02/2024, zadanie 3 (weryfikacja)

Pracujemy w języku $\textsc{Tiny}_{\mathcal{A}}$ nad typem danych rozszerzonym o operację artymetyczną `_ div2` dzielenia całkowitego przez 2 oraz o rodzaj *Array* i operacje:

$$\begin{aligned}
&\mathit{newarr}\colon \mathit{Array}\\
&\mathit{put}\colon \mathit{Array}\times \mathit{Int}\times \mathit{Int}\to \mathit{Array}\\
&\mathit{get}\colon \mathit{Array}\times \mathit{Int}\to \mathit{Int}\\
&\mathit{swap}\colon \mathit{Array}\times \mathit{Int}\times \mathit{Int}\to \mathit{Array}
\end{aligned}$$

Nośnik rodzaju *Array* to zbiór funkcji (całkowitych) z liczb całkowitych w liczby całkowite,

$$|\mathcal{A}|_{\mathit{Array}} = \mathbf{Int}\to\mathbf{Int},$$

a operacje interpretowane są jako funkcje

$$\begin{aligned}
&\mathit{newarr}_{\mathcal{A}}\colon |\mathcal{A}|_{\mathit{Array}}\\
&\mathit{put}_{\mathcal{A}}\colon |\mathcal{A}|_{\mathit{Array}}\times|\mathcal{A}|_{\mathit{Int}}\times|\mathcal{A}|_{\mathit{Int}}\to|\mathcal{A}|_{\mathit{Array}}\\
&\mathit{get}_{\mathcal{A}}\colon |\mathcal{A}|_{\mathit{Array}}\times|\mathcal{A}|_{\mathit{Int}}\to|\mathcal{A}|_{\mathit{Int}}\\
&\mathit{swap}_{\mathcal{A}}\colon |\mathcal{A}|_{\mathit{Array}}\times|\mathcal{A}|_{\mathit{Int}}\times|\mathcal{A}|_{\mathit{Int}}\to|\mathcal{A}|_{\mathit{Array}}
\end{aligned}$$

zdefiniowane następująco:

$$\begin{aligned}
&\mathit{newarr}_{\mathcal{A}}(i) = 0\\
&\mathit{put}_{\mathcal{A}}(A,i,n) = A[i\mapsto n]\\
&\mathit{get}_{\mathcal{A}}(A,i) = A(i)\\
&\mathit{swap}_{\mathcal{A}}(A,i,j) = A[i\mapsto A(j), j\mapsto A(i)]
\end{aligned}$$

dla wszystkich $i,j,n\in\mathbf{Int}$ i $A\colon \mathbf{Int}\to\mathbf{Int}$. Wyrażenie $\mathit{get}(A,e)$ będziemy jak zwykle zapisywać jako $A[e]$. Ponadto, w asercjach wykorzystujemy „predykat" $A{:}\mathit{Array}$ stwierdzajacy, że zmienna A jest rodzaju *Array* (pozostałe zmienne są rodzaju *Int*). Wyrażenia logiczne języka rozszerzamy też o oczywiste nierówności $e < e'$ (wyrażalne jako $e\le e'\wedge\neg(e=e')$) oraz $e > e'$ (wyrażalne jako $e' < e$). Dla czytelności, w formułach ciągi nierówności będziemy zapisywać w skróconej postaci, np. $j\le i\wedge i<k$ zapisując jako $j\le i<k$.

Poniżej (na odwrocie) dany jest program w tym języku.
Do celów specyfikacji wprowadzamy predykat:

$$H(a,j,k)\equiv a{:}\mathit{Array}\wedge\forall l.2*j\le l\le k\Rightarrow a[l\ \texttt{div2}]\ge a[l]$$

Należy udowodnić całkowitą poprawność programu względem podanych warunków, podając:

- niezmienniki pętli programu oraz asercje pośrednie, które „kodują" dowód poprawności częściowej w logice Hoare'a; wymagane jest podanie niezmienników γ_1 i γ_2 oraz przynajmniej asercji $\alpha_1,\alpha_2,\alpha_3$ (ale podanie innych asercji z błędami może wpłynąć na ostateczną ocenę rozwiąznia).
- anotacje [**decr ... in ... wrt ...**] dla obu pętli tak, aby (w kontekście podanych niezmienników) wynikała z nich własność stopu pętli.

```
[n > 0 ∧ a:array]
k := 1;
while [γ1:                                                    ]
   k < n do [ decr                wrt        in       ]
   (
[                                                             ]
    k := k+1;
[                                                             ]
    j := k;
[                                                             ]
    x := a[j];
[α1:                                                          ]
    while [γ2:                                                ]
       j > 1 do [ decr              wrt        in       ]
       (
[                                                             ]
        i := j div2;
[                                                             ]
        if x ≤ a[i] then
[α2:                                                          ]
           j := 1
[                                                             ]
        else (
[                                                             ]
              a := swap(a,i,j);
[α3:                                                          ]
              j := i
[                                                             ]
             )
[                                                             ]
       )
[                                                             ]
   )
[                                                             ]
[H(a,1,n)]
```

Przykładowe rozwiązanie (wersja I)

```
[n>0 ∧ a:Array]
k := 1;
while [γ1 : H(a,1,k) ∧ 1≤k≤n ]
   k < n do [ decr n-k wrt > in N ]
   (
[H(a,1,k) ∧ 1≤k<n ]
[2≤k+1≤n ∧ ∀l.((2≤l≤k+1 ∧ l≠k+1) ⇒ a[l div2]≥a[l]) ]
    k := k+1;
[2≤k≤n ∧ ∀l.((2≤l≤k ∧ l≠k) ⇒ a[l div2]≥a[l]) ]
    j := k;
[1≤j≤k ∧ 2≤k≤n ∧ ∀l.((2≤l≤k ∧ l≠j) ⇒ a[l div2]≥a[l]) ∧
  j>1 ⇒ ((2*j≤k ⇒ a[j div2]≥a[2*j]) ∧ (2*j+1≤k ⇒ a[j div2]≥a[2*j+1])) ]
    x := a[j];
[α1: 1≤j≤k ∧ 2≤k≤n ∧ ∀l.(2≤l≤k ∧ l≠j) ⇒ a[l div2]≥a[l] ∧
      j>1 ⇒ ( x=a[j] ∧ (2*j≤k ⇒ a[j div2]≥a[2*j]) ∧
                        (2*j+1≤k ⇒ a[j div2]≥a[2*j+1]) ) ]
    while [γ2: 1≤j≤k ∧ 2≤k≤n ∧ ∀l.((2≤l≤k ∧ l≠j) ⇒ a[l div2]≥a[l]) ∧
               j>1 ⇒ ( x=a[j] ∧ (2*j≤k ⇒ a[j div2]≥a[2*j]) ∧
                                 (2*j+1≤k ⇒ a[j div2]≥a[2*j+1]) ) ]
      j > 1  do [ decr j wrt > in N ]
      (
[γ2 ∧ j>1 ]
       i := j div2;
[γ2 ∧ j>1 ∧ i=j div2 ]
       if x ≤ a[i] then
[α2: γ2 ∧ j>1 ∧ i=j div2 ∧ x≤a[i] ]
[α2: 2≤k≤n ∧ ∀l.(2≤l≤k ⇒ a[l div2]≥a[l]) ]
         j := 1
[j=1 ∧ 2≤k≤n ∧ ∀l.(2≤l≤k ⇒ a[l div2]≥a[l]) ]
       else (
[γ2 ∧ j>1 ∧ i=j div2 ∧ x>a[i] ]
            a := swap(a,i,j);
[α3: 1≤i≤k ∧ 2≤k≤n ∧ ∀l.((2≤l≤k ∧ l≠i) ⇒ a[l div2]≥a[l]) ∧
      i>1 ⇒ ( x=a[i] ∧ (2*i≤k ⇒ a[i div2]≥a[2*i]) ∧
                        (2*i+1≤k ⇒ a[i div2]≥a[2*i+1]) ) ]
            j := i
[γ2 ]
           )
        )
[j=1 ∧ 2≤k≤n ∧ ∀l.((2≤l≤k ∧ l≠j) ⇒ a[l div2]≥a[l]) ]
  )
[H(a,1,k) ∧ 1≤k=n ]
[H(a,1,n)]
```

Przykładowe rozwiązanie (wersja II, tylko nieco różna)

```
[n>0 ∧ a:Array]
k := 1;
while [γ1 : H(a,1,k) ∧ 1≤k≤n ]
   k < n do [ decr n-k wrt > in N ]
   (
[H(a,1,k) ∧ 1≤k<n ]
    k := k+1;
[H(a,1,k-1) ∧ 2≤k≤n ]
    j := k;
[H(a,1,k-1) ∧ 2≤j=k≤n ]
    x := a[j];
[α1: H(a,1,k-1) ∧ 2≤j=k≤n ∧ x=a[j] ]
    while [γ2: 1≤j≤k ∧ 2≤k≤n ∧ ∀l.((2≤l≤k ∧ l≠j) ⇒ a[l div2]≥a[l]) ∧
                j>1 ⇒ ( x=a[j] ∧ (2*j≤k ⇒ a[j div2]≥a[2*j]) ∧
                                  (2*j+1≤k ⇒ a[j div2]≥a[2*j+1]) ) ]
       j > 1  do [ decr j wrt > in N ]
       (
[γ2 ∧ j>1 ]
        i := j div2;
[γ2 ∧ j>1 ∧ i=j div2 ]
        if x ≤ a[i] then
[α2: γ2 ∧ H(a,1,k) ]
          j := 1
[j=1 ∧ 2≤k≤n ∧ H(a,1,k) ]
        else (
[γ2 ∧ j>1 ∧ i=j div2 ∧ x>a[i] ]
             a := swap(a,i,j);
[α3: 1≤i≤k ∧ 2≤k≤n ∧ ∀l.((2≤l≤k ∧ l≠i) ⇒ a[l div2]≥a[l]) ∧
       i>1 ⇒ ( x=a[i] ∧ (2*i≤k ⇒ a[i div2]≥a[2*i]) ∧
                         (2*i+1≤k ⇒ a[i div2]≥a[2*i+1]) ) ]
             j := i
[γ2 ]
            )
         )
[2≤k≤n ∧ H(a,1,k) ]
   )
[H(a,1,k) ∧ k=n ]
[H(a,1,n)]
```