1 Задачите, които решихме на упражнението.

Задача 1. Да се докаже, че алгоритъмът FindMaxIndex, който на входа си непразен масив от цели числа, връща индекс на максимален елемент в този масив.

FindMaxIndex($A[1\,..\,n]$: масив от цели числа, като $n \geq 1$)

```
1 mi ← 1
2 for i ← 2 to n do
3     if A[i] > A[mi] then
4         mi ← i
5 return mi
```

Решение. Твърдим, че следното твърдение, което означаваме с $(*)$, е инвариант на цикъла:

> Всеки път, в който изпълнението е на ред 2, mi съдържа индекс на максимален елемент в подмасива $A[1\,..\,i-1]$.

Доказателство.

База. Разглеждаме първото достигане на ред 2. От една страна, $i = 2$, поради което $A[1\,..\,i-1]$ е точно $A[1\,..\,1]$, чийто единствен, а значи и максимален, елемент е $A[1]$. От друга страна, mi $= 1$, поради което mi е индекс на максимален елемент в $A[1\,..\,i-1]$. Следователно $(*)$ е в сила при първото достигане на ред 2.

Поддръжка. Нека $(*)$ е в сила за някое достигане на ред 2, което не е последното. Изпълнява се тялото на цикъла. Възможни са следните два случая:

Сл. 1. $A[i] > A[\text{mi}]$. Тогава, тъй като по предположение mi съдържа индекс на максимален елемент на $A[1\,..\,i-1]$, то $A[i]$ е максимален елемент на $A[1\,..\,i]$. На ред 4 променливата mi получава стойност i, след което i се увеличава с 1. Следователно $(*)$ е в сила при следващото достигане на ред 2.

Сл. 2. $A[i] \leq A[\text{mi}]$. Тогава, тъй като по предположение mi съдържа индекс на максимален елемент на $A[1\,..\,i-1]$, то $A[\text{mi}]$ е максимален елемент на $A[1\,..\,i]$. Стойността на mi не се променя, а i се увеличава с 1. Следователно $(*)$ е в сила при следващото достигане на ред 2.

Терминация. При последното достигане на ред 2 е изпълнено $i = n+1$, поради което $A[1\,..\,i-1]$ е точно $A[1\,..\,n]$. Доказахме, че $(*)$ е в сила за всяко достигане на ред 2. Следователно mi съдържа индекс на максимален елемент в $A[1\,..\,n]$. □

На ред 5 алгоритъмът връща mi, а както вече знаем, тази променлива съдържа индекс на максимален елемент в $A[1\,..\,n]$, с което задачата е решена.

Друг възможен инвариант на цикъла, който върши работа, е:

Всеки път, в който изпълнението е на ред 2, $A[\text{mi}]$ съдържа максимален елемент на $A[1\,..\,i-1]$.

Задача 2. Да се докаже, че алгоритъмът FindMatchingIndex, който на входа си приема сортиран масив от две по две различни цели числа, връща цяло число k, за което $1 \leq k \leq n$ и $A[k] = k$, а ако число с това свойство не съществува, връща -1.

FindMatchingIndex($A[1\,..\,n]$: сортиран масив от две по две различни цели числа)

```
1  ℓ ← 1
2  h ← n
3  while h − ℓ > 1 do
4      mid ← ⌊(ℓ+h)/2⌋
5      if A[mid] = mid then
6          return mid
7      else if A[mid] > mid then
8          h ← mid
9      else
10         ℓ ← mid
11 if A[ℓ] = ℓ then
12     return ℓ
13 else if A[h] = h then
14     return h
15 else
16     return −1
```

Решение. Твърдим, че следното твърдение, което означаваме с $(*)$, е инвариант на цикъла:

> Всеки път, в който изпълнението е на ред 3, е изпълнено, че ако съществува k, за което $1 \leq k \leq n$ и $A[k] = k$, то за всяко такова k е в сила $\ell \leq k \leq h$.

Доказателство.

База. При първото достигане на ред 3 променливите ℓ и h съдържат съответно 1 и n, така че $(*)$ очевидно е изпълнено.

Поддръжка. Нека $(*)$ е в сила за някое достигане на ред 3, което не е последното. Възможни са следните два случая:

Сл. 1. $A[\text{mid}] > \text{mid}$. Тогава, понеже A е сортиран и елементите му са два по два различни, в сила е $A[t] > t$ за всяко $\text{mid} \leq t \leq n$. Така че няма k, за което $\text{mid} \leq k \leq n$ и $A[k] = k$, откъдето заключваме, че ако има k, за което $1 \leq k \leq n$ и $A[k] = k$, то за всяко такова k е изпълнено $1 \leq k \leq \text{mid}$, но на ред 8 променливата h получава стойност mid, поради което $(*)$ остава в сила и при следващото достигане на ред 3.

Сл. 2. $A[\text{mid}] < \text{mid}$. Разсъжденията в този случай са напълно аналогични на тези в предния, като тук разликата е, че $A[t] < t$ е в сила за всяко $1 \leq t \leq \text{mid}$.

Не разглеждаме случая, в който $A[\text{mid}] = \text{mid}$, понеже от кода става ясно, че това е един от случаите, в които ред 3 се достига за последен път.

Терминация. Разглеждаме последното достигане на ред 3. Възможни са следните два случая:

Сл. 1. $h - \ell \leq 1$. Ясно е, че при всяко достигане на ред 3, което не е последното, $h - \ell > 1$, което влече $h > \ell$, поради което $\ell \leq \text{mid} \leq h$. Значи този случай се достига по един от следните два начина:

н. 1. $h - \ell = 0$. Тогава от $(*)$ получаваме, че ако в A има k, за което $A[k] = k$, то $k = \ell = h$.

н. 2. $h - \ell = 1$. Тогава от $(*)$ получаваме, че ако в A има k, за което $A[k] = k$, то $k = \ell$ или $k = h$.

Сл. 2. $h - \ell > 1$ и $A[\text{mid}] = \text{mid}$. Тогава алгоритъмът е намерил k, за което $A[k] = k$. □

Нека цикълът е приключил поради първия случай от терминацията. Както вече видяхме, ако има k, за което $1 \leq k \leq n$ и $A[k] = k$, то за всяко такова k е изпълнено $\ell \leq k \leq h$. И от кода става ясно, че редове 11, 13 и 15 изцяло покриват този случай.

Тази задача е от семестриалното контролно през 2021. Алтернативно решение на задачата можете да намерите в курса в moodle от тази година.

2 Незадължителни задачи за домашно.

Задача 3. Да се докаже, че алгоритъмът KADANE, който на входа си получава масив от цели числа, връща сумата на максимален по сума подмасив на входния.

KADANE($A[1\,..\,n]$: масив от цели числа)

```
1  gm ← 0
2  lm ← 0
3  for i ← 1 to n do
4      lm ← lm + A[i]
5      if lm < 0 then
6          lm ← 0
7      if lm > gm then
8          gm ← lm
9  return gm
```

Задача 4. Да се докаже, че алгоритъмът FINDMAJORITY, който на входа си получава непразен масив от цели числа с n елемента, в който има елемент с повече от $\lfloor \frac{n}{2} \rfloor$ срещания, връща стойността на такъв елемент в този масив.

FINDMAJORITY($A[1\,..\,n]$: масив от цели числа, удовлетворяващ посоченото по-горе условие)

```
1  c ← A[1]
2  v ← 1
3  for i ← 2 to n do
4      if v = 0 then
5          c ← A[i]
6          v ← 1
7      else
8          if A[i] = c then
9              v ← v + 1
10         else
11             v ← v − 1
12 return c
```