Зад. 1
Да се реализира стек, поддържащ min/max елемент в O(1) време и O(n) памет, където n е броя елементи в стека.

```
Struct StackWithMin:
        s ← Stack.Init()
        m ← Stack.Init()
        push(a)
        pop()
        min()
        top()
        isEmpty()

StackWithMin.push(a):
        s.push(a)
        if m.isEmpty() then
                m.push(a)
        else
                m.push(min(a, m.top()))

StackWithMin.pop():
        m.pop()
        return s.pop()

StackWithMin.min():
        if m.isEmpty() then
                return +∞
        return m.top()

StackWIthMin.top():
        return s.top()

StackWithMin.isEmpty():
        return s.isEmpty()
```

Зад. 2

Да се реализира опашка, поддържаща min/max елемент в O(1) време и O(n) памет, където n е броя елементи в опашката.

```
Struct QueueWithMin:
        s ← StackWithMin.Init()
        q ← StackWithMin.Init()
        push(a)
        pop()
        min()
        front()
        isEmpty()

QueueWithMin.push(a):
        s.push(a)

QueueWithMin.pop():
        if q.isEmpty() then
                while not s.isEmpty() do
                        q.push(s.pop())
        return q.pop()

QueueWithMin.min():
        return min(q.min(), s.min())

QueueWithMin.front():
        if q.isEmpty() then
                while not s.isEmpty() do
                        q.push(s.pop())
        return q.top()

QueueWithMin.isEmpty():
        return s.isEmpty() and q.isEmpty()
```

Зад. 3
Дадено е цяло положително число k и масив $A[1 .. n] \in \mathbb{Z}^n$. Да се намери
$\max\{L \mid \exists\, i \in \{1, .., n-k+1\} : \forall\, j \in \{i, .., i+k-1\}$ е изп. $A[j] \geq L\}$.

k=3
A=3 2 3 5 2 -1 4 3 6 2

```
Task3(A[1..n], n, k):
        if k>n then
                return -∞
        q ← QueueWithMin.Init()
        for i←1 to k
                q.push(A[i])
        L ← q.min()
        for i←k+1 to n
                q.pop()
                q.push(A[i])
                L ← max(L, q.min())
        return L
```

Зад. 4
Даден е масив от естествени чилса $A[1 .. n] \in \mathbb{N}^n$. Да се намери мажорантата на масива. Гарантирано е, че масива съдържа мажоранта.. как бихме се справили, ако не ни беше гарантирано, че има мажоранта?
Деф. Мажоранта на един масив A[1..n] е елемент от масива, срещащ се на поне $\left\lfloor \frac{n}{2} \right\rfloor + 1$ пъти (отново в масива).
5 2 5 2 5 5 5
5 5 5 5 2 2 2
5 5 5 5 2 3 2
2 5 2 5 2 5 5
2 2 2 5 5 5 5
2 3 2 5 5 5 5

```
1. Task4(A[1..n], n):
2.      currGuess ← A[1]
3.      cnt ← 1
4.      for i←2 to n
5.              if currGuess=A[i] then
6.                      cnt ← cnt+1
7.              else
8.                      cnt ← cnt-1
9.                      if cnt=0 then
10.                             currGuess ← A[i]
11.                             cnt ← 1
12.     return currGuess
```

Защо работи?
Неформално доказателство:
По условие ни е гарантирано, че имаме мажоранта.. нека я кръстим maj.
1сл. (на ред 2. currGuess ни се инициализира с елемент, който не е maj)
Тъй като maj се среща повече от половината пъти, то cnt е огр. отгоре от $\lceil \frac{n}{2} \rceil - 1$.. нека проверим най-крайния случай. Следвайки логиката на кода на $i = 2\left(\lceil \frac{n}{2} \rceil - 1\right)$ ще получим, че cnt=0 и от там currGuess ще стане на maj.. (защо?)
Друга възможност е cnt да не стигне до $\lceil \frac{n}{2} \rceil - 1$ преди да имаме случай, в който cnt намаля... тук имаме 2 възможности:
1.1сл. (cnt намаля, поради елемент различен от maj)
Тогава “отпадат” 2 числа, които не са maj и съответено maj е мажоранта на подмасива на A с размер n-2 (в който липсват тези 2 числа).. защото maj ще се среща поне $\lfloor \frac{n}{2} \rfloor + 1$ пъти в масив от n-2 елемента.. мажоранта на масив с n-2 елемента трябва да се среща $\lfloor \frac{n-2}{2} \rfloor + 1 \leq \lfloor \frac{n}{2} \rfloor + 1$
1.2сл. (cnt намаля, поради елемент равен на maj)
Тогава “отпадат” 2 числа, които са една maj и една не maj.. тогава аналогично на 1.1 maj остава мажоранта за подмасива на A с n-2 елемента... защото maj ще се среща поне $\lfloor \frac{n}{2} \rfloor + 1 - 1 = \lfloor \frac{n}{2} \rfloor = \lfloor \frac{n-2}{2} \rfloor + 1$ в масив от n-2 елемента
2сл. (на ред 2. currGuess ни се инициализира с елемент, който е равен на maj)
Тъй като maj се среща поне $\lfloor \frac{n}{2} \rfloor + 1$ пъти, то ако всички maj са най-вляво на масива ще имаме, че cnt е ограничено отдолу от $\lfloor \frac{n}{2} \rfloor + 1$... не ка разгледаме най-крайния вариант.. Следвайки логита на програмата, ще инкрементираме cnt поне $\lfloor \frac{n}{2} \rfloor + 1$ на брой пъти, след което ще го декрементираме най-много $\lceil \frac{n}{2} \rceil - 1$ пъти, т.е cnt няка как да падне до 0 ⇒ currGuess ще е maj при приключване на програмата
Нека да погледнем не толкова крайния случай.. тоес cnt да декрементира преди cnt да е стигнало $\lfloor \frac{n}{2} \rfloor + 1$.. тогава със сигурност сме декрементирали заради елемент, който не е maj.. тоест “неутрализираме” 2 числа - едното maj, другто ne maj.. аналогично на 1.2сл.

Зад. 5
Да се провери дали ориентиран граф G=(V, E) има цикъл.

```
hasCycleFrom(G=(V, E), n, colors[1..n], startVertex):
        if colors[startVertex]=black then
                return false
        s ← Stack.Init()
        s.push(startVertex)
        while not s.isEmpty() do
                q ← s.top()
                if color[q]=gray then
                        color[q] ← black
                        s.pop()
                else // white -- cannot be black
                        colors[q] ← gray
                        for each (q, p)∈E do
                                if colors[p]=white then
                                        s.push(p)
                                else if colors[p]=gray then
                                        return true
        return false

hasCycle(G=(V, E)):
        n ← |V|
        colors[1..n] ← [white, …, white]
        for each v∈V do
                if hasCycleFrom(G, n, colors, v) then
                        return true
        return false
```

Зад. 6
Да се провери дали свързан неориентиран граф е двуделен.

to be continued