

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ  
ТАРАСА ШЕВЧЕНКА  
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Звіт до лабораторної роботи №1.1  
на тему  
**«Списки»**

Студента 3 курсу ФКНК  
групи ТТП-31  
Корнієнка Олександра

Київ-2023

## **Зміст**

<b>Вступ .....</b>	<b>3</b>
<b>Код програми .....</b>	<b>3</b>
<b>Тести .....</b>	<b>4</b>
1. Тест з унікальними числами: .....	4
2. Тест з повторюваними числами:.....	4
3. Тест зі зростаючою послідовністю: .....	5
4. Тест зі спадаючою послідовністю: .....	5
5. Тест з одним числом: .....	5
6. Тест з дублюванням: .....	5
<b>Результати тестів .....</b>	<b>6</b>

# Вступ

Модуль 1.1

Розділ 1. Списки

Варіант 9

Знайти N найменших елементів списку та сформувати з них новий список.

## Код програми

```
import Data.List (elemIndex, sort)

findSmallestN :: (Ord a) => Int -> [a] -> [a]
findSmallestN n array = take n (sort array)

findIndexes :: (Eq a) => [a] -> [a] -> [Int]
findIndexes array1 array2 = mapMaybe (`elemIndex` array2) array1
  where
    mapMaybe _ [] = []
    mapMaybe f (x : xs) =
      case f x of
        Just index -> index : mapMaybe f xs
        Nothing -> mapMaybe f xs

getElementAtIndex :: (Num a) => Int -> [a] -> a
getElementAtIndex _ [] = -1
getElementAtIndex index (a : rest)
  | index < 0 = -1
  | index == 0 = a
  | otherwise = getElementAtIndex (index - 1) rest

main :: IO ()
main = do
```

```

content <- lines <$> readFile "input.txt"
let n = read (head content) :: Int
let numbers = map read (words (content !! 1)) :: [Int]
-- 0 show input
putStrLn "input:"
print n
print numbers
-- 1 find all smallest numbers of n
let arrayOfSmallest = findSmallestN n numbers
-- 2 find first indexes for those smallest and push to array
let arrayOfIndexes = findIndexes arrayOfSmallest numbers
-- 3 sort arrayOfIndexes
let sortedArrayOfIndexes = sort arrayOfIndexes
-- 4 substitute indexes with values
let result = map (`getElementAtIndex` numbers) sortedArrayOfIndexes
-- 5 show result
putStrLn "outcome:"
print result

```

## Тести

Тести по чергово вносяться у файл input.txt, де перший рядок відповідає за константу N, а другий рядок за масив цілих чисел

### 1. Тест з унікальними числами:

- Вхід:

4

9 2 6 1 4 8

- Вихід: [2, 6, 1, 4]

### 2. Тест з повторюваними числами:

- Вхід:

3

5 2 7 2 4 6

- Вихід: [2, 2, 4]

### 3. Тест зі зростаючою послідовністю:

- Вхід:

2

1 2 3 4 5

- Вихід: [1, 2]

### 4. Тест зі спадаючою послідовністю:

- Вхід:

3

9 8 7 6 5 4

- Вихід: [6,5,4]

### 5. Тест з одним числом:

- Вхід:

1

3

- Вихід: [3]

### 6. Тест з дублюванням:

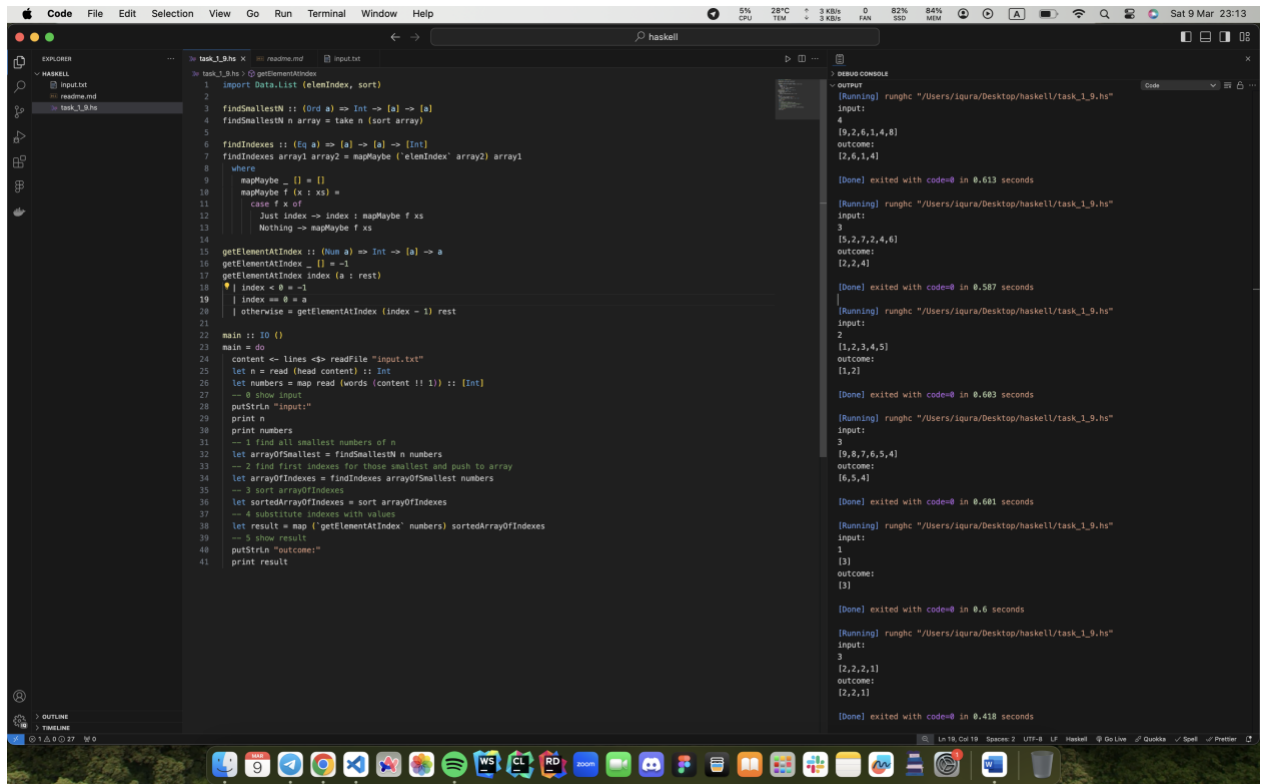
- Вхід:

3

2 2 2 1

- Вихід: [2,2,1]

# Результати тестів



The screenshot shows a Haskell code editor with a file named `task_1_9.hs`. The code defines functions for finding the smallest numbers and their indices in a list. The `main` function reads input from `input.txt` and prints the results.

```
1 import Data.List (elemIndex, sort)
2
3 findSmallestN :: (Ord a) => Int -> [a] -> [a]
4 findSmallestN n array = take n (sort array)
5
6 findIndexes :: (Eq a) => [a] -> [a] -> [Int]
7 findIndexes array1 array2 = mapMaybe (\elemIndex' array2) array1
8   where
9     mapMaybe _ [] = []
10    mapMaybe f (x : xs) =
11      case f x of
12        Just index -> index : mapMaybe f xs
13        Nothing -> mapMaybe f xs
14
15 getElementAtIndex :: (Num a) => Int -> [a] -> a
16 getElementAtIndex _ [] = -1
17 getElementAtIndex index (a : rest)
18   | index < 0 = -1
19   | index == 0 = a
20   | otherwise = getElementAtIndex (index - 1) rest
21
22 main :: IO ()
23 main = do
24   content <- lines <$> readFile "input.txt"
25   let n = read (head content) :: Int
26   let numbers = map read (words (content !! 1)) :: [Int]
27   - 0 show input
28   putStrLn "input:"
29   print n
30   print numbers
31   - 1 find all smallest numbers of n
32   let arrayOfSmallest = findSmallestN n numbers
33   - 2 find first indexes for those smallest and push to array
34   let arrayOfIndexes = findIndexes arrayOfSmallest numbers
35   - 3 sort arrayOfIndexes
36   let sortedArrayOfIndexes = sort arrayOfIndexes
37   - 4 substitute indexes with values
38   let result = map (\getElementAtIndex' numbers) sortedArrayOfIndexes
39   - 5 show result
40   putStrLn "outcome:"
41   print result
```

The right pane shows the output of the program for five different test cases:

```
[Running] runghc "/Users/igora/Desktop/haskell/task_1_9.hs"
Input:
4
[9,2,6,1,4,8]
outcome:
[2,6,1,4]

[Done] exited with code=0 in 0.613 seconds

[Running] runghc "/Users/igora/Desktop/haskell/task_1_9.hs"
Input:
3
[5,2,7,2,4,6]
outcome:
[2,2,4]

[Done] exited with code=0 in 0.587 seconds

[Running] runghc "/Users/igora/Desktop/haskell/task_1_9.hs"
Input:
2
[1,2,3,4,5]
outcome:
[1,2]

[Done] exited with code=0 in 0.603 seconds

[Running] runghc "/Users/igora/Desktop/haskell/task_1_9.hs"
Input:
3
[9,8,7,6,5,4]
outcome:
[6,5,4]

[Done] exited with code=0 in 0.601 seconds

[Running] runghc "/Users/igora/Desktop/haskell/task_1_9.hs"
Input:
1
[3]
outcome:
[3]

[Done] exited with code=0 in 0.6 seconds

[Running] runghc "/Users/igora/Desktop/haskell/task_1_9.hs"
Input:
3
[2,2,2,1]
outcome:
[2,2,1]

[Done] exited with code=0 in 0.418 seconds
```