

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

Звіт до лабораторної роботи №2  
на тему  
**«Списки. Додаткові задачі»**

Студента 3 курсу ФКНК  
групи ТТП-31  
Корнієнка Олександра

Київ-2023

## **Зміст**

<b><i>Вступ</i></b> .....	<b>3</b>
<b><i>Код програми</i></b> .....	<b>3</b>
<b><i>Тести</i></b> .....	<b>5</b>
Тест 1.....	5
Тест 2.....	5
Тест 3.....	5
Тест 4.....	5
Тест 5.....	5
<b><i>Результати тестів</i></b> .....	<b>7</b>

# Вступ

Модуль 2

Розділ 2. Списки. Додаткові задачі

Варіант 25

Розбити заданий список на кілька підсписків, записуючи, за можливості, у перший і останній по  $p$  1 елементів, потім, за можливості, у другий і передостанній – по  $p$  2 елементів і т.д. Тут через  $p$  і позначено  $i$ -е просте число у списку всіх простих чисел.

## Код програми

```
printResult(Array):-
    length(Array, Length),
    Length == 0 -> write("");
    length(Array, Length),
    write(Array),
    write(' Length: '),
    write(Length),
    nl.

isDivisible(X, Y):-
    0 is X mod Y,!.

isPrime(2):- true,!.
isPrime(X):- X < 2,!,false.
isPrime(X):-
    Limit is floor(sqrt(X)),
    \+ (between(2, Limit, Y), X \= Y, isDivisible(X, Y)).

findNextPrime(Num, Result):-
    Num1 is Num + 1,
    (isPrime(Num1) -> Result = Num1;
    findNextPrime(Num1, Result)).

subString(Array, Divider, BeforeFirst, Between, AfterLast):-
```

```
length(Array, ArrayLength),  
FirstDivider is Divider,  
LastDivider is ArrayLength - Divider,  
split_at(FirstDivider, Array, BeforeFirst, Rest),  
RemainingLength is LastDivider - FirstDivider,  
split_at(RemainingLength, Rest, Between, AfterLast).
```

```
split_at(0, L, [], L).
```

```
split_at(N, [X|Xs], [X|Ys], Zs):-
```

```
    N > 0,  
    N1 is N - 1,  
    split_at(N1, Xs, Ys, Zs).
```

```
recursiveSubString(Array, Divider):-
```

```
    length(Array, ArrayLength),  
    Divider * 2 > ArrayLength -> printResult(Array);  
    subString(Array, Divider, BeforeFirst, Between, AfterLast),  
    printResult(BeforeFirst),  
    findNextPrime(Divider, Prime),  
    recursiveSubString(Between, Prime),  
    printResult(AfterLast).
```

```
run :-
```

```
    Array=[],  
    write('input: '),  
    writeln(Array),  
    write('outcome: '),nl,  
    length(Array,ArrayLength),  
    (ArrayLength == 0 -> write([]), nl;  
    findNextPrime(0, Prime),  
    recursiveSubString(Array, Prime)).
```

## Тести

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

### Тест 1

Вхід: []

Вихід []

### Тест 2

Вхід: [1]

Вихід: [1]

### Тест 3

Вхід: [1, 2, 3]

Вихід: [1,2,3]

### Тест 4

Вхід: [1, 2, 3, 4]

Вихід: [1,2] [3,4]

### Тест 5

Вхід: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]

Вихід:

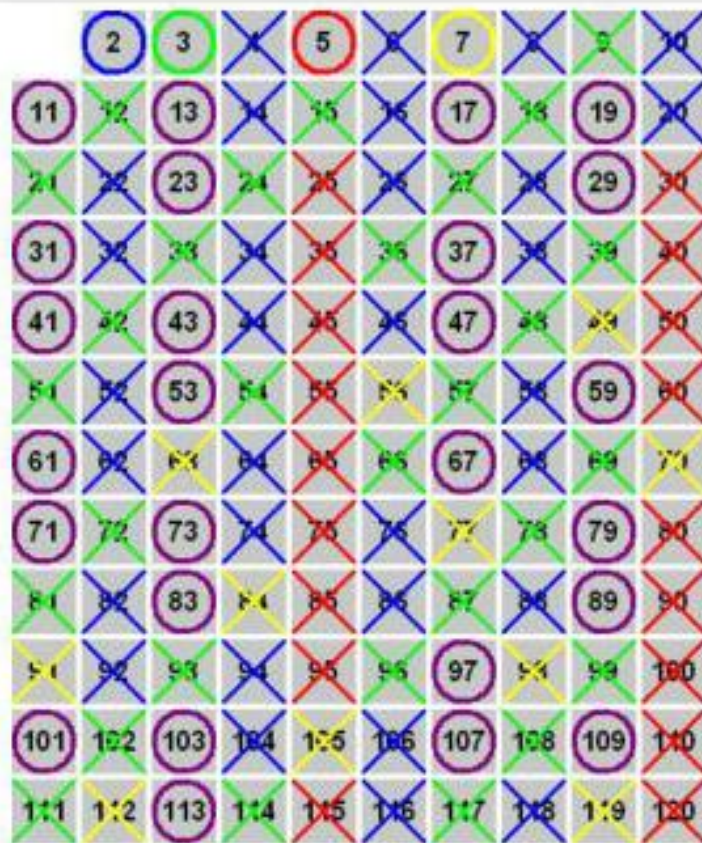
[0,1] [2,3,4] [5,6,7,8,9] [10,11,12,13,14,15,16] [17,18,19,20,21,22,23,24,25,26,27]

[28,29,30,31,32,33,34,35,36,37,38,39,40]

[41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59]

[60,61,62,63,64,65,66,67,68,69,70,71,72] [73,74,75,76,77,78,79,80,81,82,83]

[84,85,86,87,88,89,90] [91,92,93,94,95] [96,97,98] [99,100]



# Prime numbers

2    3    5    7  
11    13    17    19  
23    29    31    37  
41    43    47    53  
59    61    67    71  
73    79    83    89  
97    101    103    107  
109    113

# Результати тестів

The screenshot shows a Haskell code editor with the following code in `task_2_25.hs`:

```
33 split_at(N, [X|Xs], [Y|Ys], Zs):-
34   split_at(N1, Xs, Ys, Zs).
35
36 recursiveSubString(Array, Divider):-
37   length(Array, ArrayLength),
38   Divider < 2 > ArrayLength -> printResult(Array);
39   recursiveSubString(Array, Divider - 1, BeforeFirst, BeforeLast, AfterLast).
```

The terminal output shows the results of running the program for various inputs:

```
7- run.
input: []
outcome: []
true.
7- ["Users/Igor/Desktop/haskell/task_2_25.pl"].
true.

7- run.
input: [1]
outcome: [1]
true.
7- ["Users/Igor/Desktop/haskell/task_2_25.pl"].
true.

7- run.
input: [1,2,3]
outcome: [1,2,3]
true.
7- ["Users/Igor/Desktop/haskell/task_2_25.pl"].
true.

7- run.
input: [1,2,3,4]
outcome: [1,2]
true.
7- ["Users/Igor/Desktop/haskell/task_2_25.pl"].
true.

7- run.
input: [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100]
outcome: [1,2]
true.
7- ["Users/Igor/Desktop/haskell/task_2_25.pl"].
true.
```

The editor also shows a file explorer on the left with files like `task_1_52.pl`, `task_2_25.pl`, and `task_2_25.hs`. The bottom status bar indicates the current line and column: `Ln 39, Col 32`.