

Міністерство освіти і науки, молоді та спорту України
Національний університет “Львівська політехніка”

Кафедра ЕОМ



Звіт

з домашнє завдання №25.1 з дисципліни:

“Алгоритми та моделі обчислень”

Варіант: № 24.

Виконав:

ст. групи КІ-203

Ширий Богдан Ігорович

Перевірів:

ст. викладач кафедри ЕОМ

Козак Назар Богданович

ЗАВДАННЯ:

УМОВА:

Застосовуючи парадигму функційного програмування скласти програму мовою Haskell, яка виконує імплементацію швидкого сортування без використання готових бібліотечних реалізацій.

ВИБІР ВАРІАНТУ:

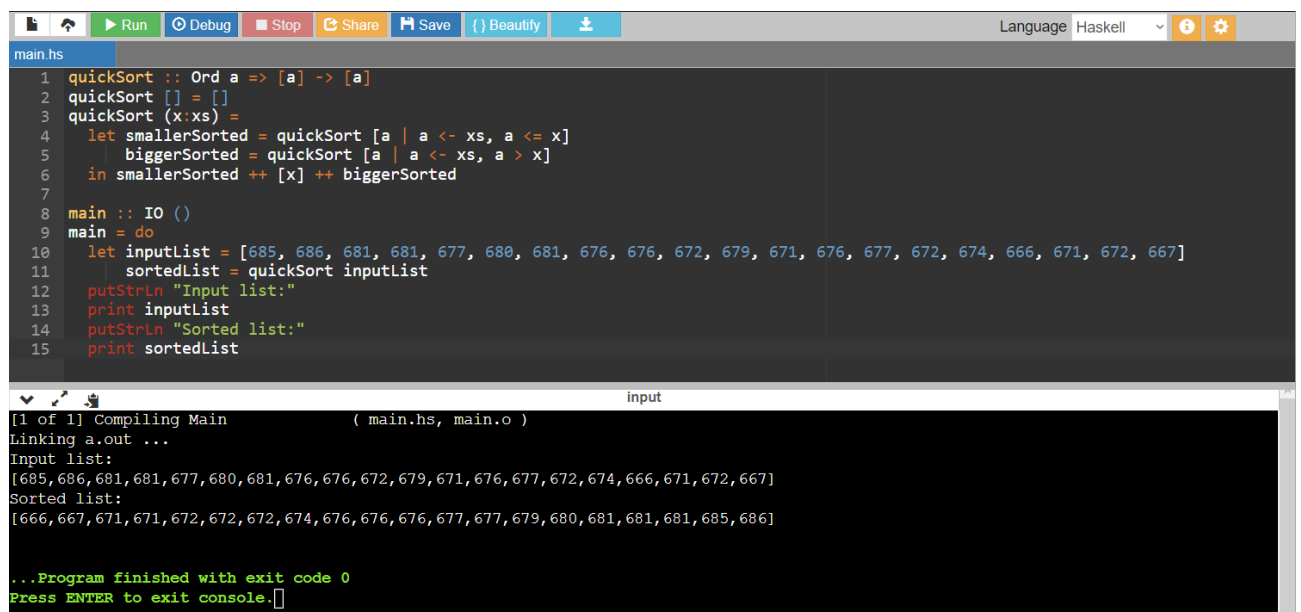
$$(N_{\text{ж}} + N_{\text{г}} + 1) \% 30 + 1 = (24 + 3 + 1) \% 30 + 1 = 28 \% 30 + 1 = 29,$$

де: $N_{\text{ж}}$ – порядковий номер студента в групі, а $N_{\text{г}}$ – номер групи.

Отож, мій шуканий варіант – це 685, 686, 681, 681, 677, 680, 681, 676, 676, 672, 679, 671, 676, 677, 672, 674, 666, 671, 672 та 667.

ВИКОНАННЯ:

Написав код на мові програмування **Haskell** та навів його компіляцію у онлайн ресурсі ([ПОСИЛАННЯ](#)) на рисунку 1.



```
main.hs
1 quickSort :: Ord a => [a] -> [a]
2 quickSort [] = []
3 quickSort (x:xs) =
4   let smallerSorted = quickSort [a | a <- xs, a <= x]
5       biggerSorted = quickSort [a | a <- xs, a > x]
6   in smallerSorted ++ [x] ++ biggerSorted
7
8 main :: IO ()
9 main = do
10  let inputList = [685, 686, 681, 681, 677, 680, 681, 676, 676, 672, 679, 671, 676, 677, 672, 674, 666, 671, 672, 667]
11      sortedList = quickSort inputList
12      putStrLn "Input list:"
13      print inputList
14      putStrLn "Sorted list:"
15      print sortedList

input
[1 of 1] Compiling Main             ( main.hs, main.o )
Linking a.out ...
Input list:
[685,686,681,681,677,680,681,676,676,672,679,671,676,677,672,674,666,671,672,667]
Sorted list:
[666,667,671,671,672,672,672,674,676,676,676,677,677,679,680,681,681,681,685,686]

...Program finished with exit code 0
Press ENTER to exit console.
```

Рис. 1. Результат роботи написаного коду.

При описі програми не використовував готових бібліотечних реалізацій, а використав такі парадигми функційного програмування:

- **Незмінність даних:** У функції **quickSort** не змінював вхідний список, а замість цього створив нові списки, які містять відсортовані елементи.
- **Рекурсія:** Функція **quickSort** використовує рекурсію для поділу вхідного списку на менші частини та сортування їх.

- **декларативне програмування:** Описав алгоритм швидкого сортування у вигляді визначення функції **quickSort**, а не явно крок за кроком наводив деталі роботи алгоритму.
- **Використання функцій вищих порядків:** У функції **quickSort** використав функціональні можливості мови, такі як спискові зрізи та умовні вирази, для фільтрації елементів і створення нових списків. Також, використав лямбда-функції для визначення умов для фільтрації.

У лістингу 1 навів код програми, що вирішує поставлене завдання.

Лістинг 1. Код програми написаної на мові Haskell.

```
quickSort :: Ord a => [a] -> [a]
quickSort [] = []
quickSort (x:xs) =
    let smallerSorted = quickSort [a | a <- xs, a <= x]
        biggerSorted = quickSort [a | a <- xs, a > x]
    in smallerSorted ++ [x] ++ biggerSorted

main :: IO ()
main = do
    let inputList = [685, 686, 681, 681, 677, 680, 681, 676, 676, 672, 679,
671, 676, 677, 672, 674, 666, 671, 672, 667]
        sortedList = quickSort inputList
    putStrLn "Input list:"
    print inputList
    putStrLn "Sorted list:"
    print sortedList
```