Упражнения: Алгоритми върху линейни структури от данни – Стекове и опашки – 3

Problem 1. Обръщане на числа със стека

Напишете програма, която чете **N цели числа** от конзолата и **ги обръща в обратен на въвеждането ред, чрез стек**.

Използвайте **Stack<int>** класа от .NET Framework. Просто поставете (**put)** въвежданите числа в стека и ги вземете (**pop)** после от стека.

Примери

Вход	Изход
`1 2 3 4 5`	`5 4 3 2 1`
`1`	`1`
`(empty)`	`(empty)`
`1 -2`	`-2 1`

Problem 2. Изчислете редицата с опашка

Дадена е следната последователност от числа:

- $S_1 = N$
- $S_2 = S_1 + 1$
- $S_3 = 2*S_1 + 1$
- $S_4 = S_1 + 2$
- $S_5 = S_2 + 1$
- $S_6 = 2*S_2 + 1$
- $S_7 = S_2 + 2$
- …

Използвайте класа **Queue<T>** и напишете програма, която извежда първите 50 члена за даденото N

Примери:

Вход	Изход
`2`	`2, 3, 5, 4, 4, 7, 5, 6, 11, 7, 5, 9, 6, …`
`-1`	`-1, 0, -1, 1, 1, 1, 2, …`
`1000`	`1000, 1001, 2001, 1002, 1002, 2003, 1003, …`

Problem 3. * Редица N → M

Дадени са числата n и m и следните операции:

a) n → n + 1
b) n → n + 2
c) n → n * 2

Напишете програма, която **намира най-късата редица от операции** от списъка по-долу, който **започва от n и завършва в m**. Ако съществуват няколко най-къси редици, намерете първата от тях.

Примери:

Вход	Изход
`3 10`	`3 -> 5 -> 10`
`5 -5`	`(няма решение)`
`10 30`	`10 -> 11 -> 13 -> 15 -> 30`

Подсказка: използвайте **опашка** и следващия алгоритъм:

1. създайте опашка от числа
2. опашка ← n
3. докато (опашката не е празна)
    1. опашка → e
    2. ако (e < m)
        i. опашка ← e + 1
        ii. опашка ← e + 2
        iii. оопшка ← e * 2
    3. ако (e == m) Print-Solution; край

С по-горния алгоритъм ще намерите решение, или ще откриете, че то не съществува. Той не може да отпечата числата, включващи редицата n → m.

За да отпечатате редицата от стъпки, за да достигне м, започвайки от n, ще трябва да запазите също и предишния елемент. Вместо с опашка от числа, използвайте опашка от елементи. Всеки елемент ще запази число и указател към предишния елемент. Промените в алгоритъма са примерно такива:

Алгоритъм Find-Sequence (n, m):

1. създайте опашка от елемент {стойност, предходен_елемент }
2. опашка ← { n, **null** }
3. докато (опашката не е празна)
    1. опашка → елемент
    2. ако (елемент.стойност < m)
        i. опашка ← { елемент.стойност + 1, елемент}
        ii. queue ← { елемент.стойност + 2, елемент }
        iii. queue ← { елемент.стойност * 2, елемент }
    3. ако (елемент.стойност == m) Print-Solution; **край**

Алгоритъм Print-Solution (item):

1. докато (елемента не е null)
    1. отпечатай елемент.стойност
    2. елемент=елемент.предходен_елемент

Министерство на образованието и науката (МОН)

- Настоящият курс (презентации, примери, задачи, упражнения и др.) е разработен за нуждите на Национална програма "**Обучение за ИТ кариера**" на МОН за подготовка по професия "Приложен програмист".

- Курсът е базиран на учебно съдържание и методика, предоставени от **фондация "Софтуерен университет"** и се разпространява под **свободен лиценз CC-BY-NC-SA** (Creative Commons Attribution-Non-Commercial-Share-Alike 4.0 International).