Контролно 1 по Структури от данни и програмиране, спец. КН, вариант А

24.11.2022 г.

За решението на задачите да се използват типовете, които са посочени в условията. Решения, които обработват некоректно паметта или не спазват добрите практики от ООП и СДП, се оценяват с 0 точки.

Елементите на едносвързан списък се представят чрез следния шаблон на структура:

```
template <typename T>
struct Node {
      T value;
      Node<T>* next;
};
```

Задача 1. (1,5 т.) Нека е даден едносвързан списък. Всеки затворен интервал от валидни 0-базирани индекси [i, j] представя еднозначно негов подсписък, състоящ се от последователните елементи с индекси k ∈ [i, j]. Да се реализира шаблон на функция `reverseSublists(<тип> list, <тип> intervals)`, която приема указател към първи елемент на едносвързан списък `list` с елементи от тип T и указател към първи елемент на едносвързан списък `intervals` с елементи от подходящ тип, представящ затворен интервал от 0-базирани индекси, и **модифицира** списъка `list` като последователно обръща огледално реда на елементите във всеки от подсписъците, представени чрез интервалите в `intervals`. Ако някой интервал съдържа невалидни индекси в `list`, да се хвърли подходящо изключение (`exception`).

Пример:

`intervals`	`[1, 3] → [2, 5] → [5, 6]`
`list`	`11 → 4 → 3 → 7 → 13 → 4 → 5`
след обръщане на [1, 3]	`11 → 7 → 3 → 4 → 13 → 4 → 5`
след обръщане на [2, 5]	`11 → 7 → 4 → 13 → 4 → 3 → 5`
след обръщане на [5, 6]	`11 → 7 → 4 → 13 → 4 → 5 → 3`

Задача 2. (1,5 т.) Списък от цели числа се нарича **изцяло балансиран**, ако се състои от поне 3 елемента и всички елементи без първия и последния са равни на сбора от съседните си два елемента и **изцяло небалансиран**, ако в него няма нито един такъв елемент. Да се реализира функция `makeTotal`, която приема като параметър указател към първия елемент на едносвързан списък от цели числа и връща `true`, ако списъкът е изцяло балансиран. Ако списъкът не е изцяло балансиран, функцията да връща `false` след като изтрие последователно от списъка от началото към края елементи, които изпълняват свойството за балансираност, докато списъкът стане изцяло небалансиран. За реализацията НЕ се допуска използването на допълнителни помощни структури от данни (т.е. външната (auxiliary) сложност по памет е O(1)).

Примери:

1)	`1 → 4 → -5 → -1 → 4`	`true`
2)	`10 → 5 → -5 → -15 → 2`	`false`
	`10 → –5 → -15 → 2`	
	`10 → -15 → 2`	*(списъкът вече е изцяло небалансиран)*

Задача 3. (1 т.) Казваме, че дадена дума може да се прочете в стек от символи, ако можем да я получим при последователно изваждане на символи от стека като избирателно пропуснем някои от тях. Да се реализира функция `readAndDelete`, която по подаден стек от символи, представен чрез `std::stack` и дума (**word**) от тип `std::string`, връща като резултат дали думата може да се прочете в стека. Функцията да модифицира стека като изключи от него всички открити символи от **word** и да запазва останалите символи в същия ред.

Примери:

word	**стек**	**резултат**		**word**	**стек**	**резултат**
mom	~~m~~ f ~~o~~ o ~~m~~	true f o		car	~~c~~ c d ~~a~~ y	false c d y

Контролно 1 по Структури от данни и програмиране, спец. КН, вариант Б

24.11.2022 г.

За решението на задачите да се използват типовете, които са посочени в условията. Решения, които обработват некоректно паметта или не спазват добрите практики от ООП и СДП, се оценяват с 0 точки.

Елементите на едносвързан списък се представят чрез следния шаблон на структура:

```
template <typename T>
struct Node {
      T value;
      Node<T>* next;
};
```

Задача 1. (1,5 т.) Да се реализира шаблон на функция `rotateSublists(<тип> list, <тип> intervals)`, която приема указател към първи елемент на едносвързан списък `list` с елементи от тип T и указател към първи елемент на едносвързан списък `indexPairs` с елементи от подходящ тип, представящ наредени двойки (i, j) от 0-базирани индекси, и **модифицира** списъка `list` като последователно изтрива елемент с индекс j и го вмъква преди елемента с индекс i. Ако някоя двойка съдържа невалидни индекси в `list`, да се хвърли подходящо изключение (`exception`).

Пример:

`indexPairs`	`(1, 3) → (2, 5) → (5, 6)`
`list`	`11 → 4 → 3 → 7 → 13 → 4 → 5`
след завъртане на (1, 3)	`11 → 7 → 4 → 3 → 13 → 4 → 5`
след завъртане на (2, 5)	`11 → 7 → 4 → 4 → 3 → 13 → 5`
след завъртане на (5, 6)	`11 → 7 → 4 → 4 → 3 → 5 → 13`

Задача 2. (1,5 т.) Списък от цели числа се нарича **изцяло последователен**, ако се състои от поне 2 елемента и всеки два негови последователни елемента са съседни числа и **изцяло непоследователен**, ако никои два последователни негови елемента не са съседни числа. Да се реализира функция `makeTotal`, която приема като параметър указател към първия елемент на едносвързан списък от цели числа и връща `true`, ако списъкът е изцяло последователен. Ако списъкът не е изцяло последователен, функцията да връща `false` след като изтрие последователно от списъка от началото към края втория от всяка двойка елементи, които са съседни числа, докато списъкът стане изцяло непоследователен. За реализацията НЕ се допуска използването на допълнителни помощни структури от данни (т.е. външната (auxiliary) сложност по памет е O(1)).

Примери:

1)	`2 → 1 → 2 → 3 → 4 → 3`	`true`
2)	`1 → ~~2~~ → 3 → 4 → 2 → 5`	`false`
	`1 → 3 → ~~4~~ → 2 → 5`	
	`1 → 3 → ~~2~~ → 5`	
	`1 → 3 → 5`	*(списъкът вече е изцяло непоследователен)*

Задача 3. (1 т.) Казваме, че дадена дума може да се прочете в опашка от символи, ако можем да я получим при последователно изваждане на символи от опашката като избирателно пропуснем някои от тях. Да се реализира функция, която по подадена опашка от символи, представена чрез `std::queue` и дума (**word**) от тип `std::string`, връща като резултат дали думата може да се прочете в опашката. Функцията да модифицира опашката като изключи от нея всички открити символи от **word** и да запазва останалите символи в **обратен** ред.

Примери:

word	**опашка**	**резултат**		**word**	**опашка**	**резултат**
mom	~~m~~ f ~~o~~ o ~~m~~	true o f		car	~~c~~ c d ~~a~~ y	false y d c