Zadania do rozwiązania

Zadanie 1.

Jednym z głównych celów korzystania ze inteligentnych wskaźników jest automatyczne zwalnianie przez nie pamięci. Popraw program w pliku zad1.cpp, tak by nie korzystał z surowych wskaźników poprzez zamienienie ich na wskaźniki inteligentne. W celu uniknięcia wycieków pamięci.

Zadanie 2.

Stwórz shared_ptr sharedPtr o dowolnej wskazywanej wartości dowolnego typu. Na jego podstawie utwórz weak_ptr weakPtr, a za pomocą funkcji use_count() oraz expired() zbadaj ich właściwości. Zwróć uwagę, które funkcje są dostępne dla danych typów wskaźników. Następnie funkcją lock() zbadaj informacje przechowywane przez sharedPtr, a na koniec zwolnij weakPtr i ponownie zbadaj informacje przechowywane przez sharedPtr.

W celu ułatwienia odczytu stanu wskaźników skorzystaj z manipulatora std::boolalpha.

Zadanie 3.

Korzystając ze smart pointerów zaproponuj własną implementację listy jednokierunkowej. Lista ma przechowywać dwie dane: data1 i data2(typy danych mają być różne). Dodatkowo lista powinna posiadać informację o długości listy i być uaktualniana wraz z działaniem odpowiednich metod. Zaimplementuj metody:

- void AddElement - dodająca nowy element na koniec listy
- void WriteOut - wypisująca całą listę
- void DeleteSpecifiedElement - usuwająca węzeł o określonych parametrach
- void DeleteList - usuwająca całą listę
- int CurrentLenght - zwracającą obecną długość listy

Przed implementacją zastanów się nad wyborem odpowiedniego wskaźnika (Podpowiedź: przy użyciu niektórych metod możliwe że będą wskaźniki na elementy listy).