Zestaw 4

1. W klasie `TString` (kompletny kod w serwisie Pegaz) proszę dopisać:
   a. `char* operator[](const char* s)`, który będzie sprawdzał, czy w danym obiekcie `TString` znajduje się zadany podciąg znaków `s` i jeśli tak, zwracał adres jego początku, a jeśli nie, zwracał `end()`; tak samo dla `char& operator[](char c)`.
   b. operator++ (obie wersje, pre- i post-), który będzie zamieniał (podwyższał) kody ASCII znaków znajdujących się w tablicy `ptr` o jeden w górę
   c. operator<=>

   W programie proszę zademonstrować przypadki użycia powyższych.
2. Napisz klasę (np. o nazwie Konwerter), która będzie miała jedną składową typu `unsigned long` (albo: `size_t` czy `size_type`). Klasa ta powinna mieć:
   a. operatory konwersji z typów `char`, `const char*`, `std::string`, które będą wyliczać „liczbę” na zasadzie char – kod ASCII, łańcuch znakowy – suma kodów ASCII poszczególnych znaków w łańcuchu i wartość tę przypisywać składowej
   b. operatory >> oraz << za pomocą których będzie można do obiektu typu Konwerter wczytać wartość całkowitą, char lub łańcuch znakowy, albo wypisać na ekran wartość składowej (całkowitej) tego obiektu

   Proszę zademonstrować również działanie w ramach programu.
3. Napisać kod dla klasy `TArr` (jak niżej) realizujący idiom **C**opy-**O**n-**W**rite (bardzo podobny przykład jest na slajdach „wykładowych”). Proszę zaimplementować konstruktory: zwykły, kopiujący, przenoszący, operatory= kopiujący i przenoszący, destruktor oraz dowolną metodę, która właśnie „coś zmienia” w zasobie, czyli tablicy do wskaźnika buf i powodującą wykonanie „głębokiej kopii”. Proszę nie zapomnieć o kilku liniach testowego kodu, również metodzie, dzięki której można sprawdzić ile obiektów współdzieli zasób w danym momencie.

```
class TArr {
    private:
        struct InnerArray {
            std::size_t len{0};
            std::size_t ref{0};
            int *buf{nullptr};
        } *ptr{nullptr};
};
```

4. Napisz klasę `TSmartPtr`, w której przeładujesz operatory -> oraz *, zaprezentuj w programie. Przykład takiego kodu jest pokazany na slajdach „wykładowych”.
5. Dla prostej klasy (choćby takiej jak klasa `A` ze składnikiem `int i;` na slajdach „wykładowych”) napisz przeciążone operatory `++`, `--` (pre- i post-), jednoargumentowe `+` i `-` oraz dwuargumentowe `+` i `-`. Następnie wykonaj dyskusję (prezentację) ile maksymalnie znaków `+` (`-`) można postawić przed lub za obiektem typu `A`, żeby kod był nadal ważny i działający. Jest to zależne od implementacji operatorów (w szczególności tego, czy zwracany typ jest z `const` czy bez).