Zestaw 4

1. Mamy klasę:

```
class TKlasa {
public:
          TKlasa(const char* c); // zmień to na definicję z inicjalizacją
private:
          string str;
};
```

   Proszę do niej dopisać:
   a. const char* operator[](const char* s) const, który będzie sprawdzał, czy w danym obiekcie TKlasa znajduje się zadany podciąg znaków s i jeśli tak, zwracał adres jego początku, a jeśli nie, zwracał nullptr.
   b. operator<=>

   W programie proszę **zademonstrować** przypadki użycia powyższych.

2. Mamy klasę:

```
class TComplex {
public:
          TComplex(double re, double im); // zamień na definicję z inicjalizacją

private:
          double re{0}, im{0};
};
```

   Proszę dopisać do niej:

   a) operator funkcyjn, żeby zapis obiekt(2,3) tworzył nowy obiekt (zwracany z niego przez wartość)
   b) przeciążyć operatory + i –
   c) napisać metody conj() zwracające sprzężenie zespolonego i abs() zwracającą wartość
   d) przeciążyć operatory * i / (sprawdzić w źródłach jak to jest dla liczby zespolonej)
   e) napisać operator<< żeby wypisywał na ekranie na przykład: (2,3)

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).