Wspólne wytyczne do wszystkich tematów projektów

Klasa – jeśli to wskazane i/lub konieczne - powinna zawierać:

- Konstruktory: domyślny i z parametrami
- Konstruktor kopiujący
- Destruktor
- Zredefiniowany operator przypisania
- Przeciążone operatory biblioteki *iostream*
- Funkcje udostępniające wartości parametrów obiektów
- Pole przechowujące liczbę istniejących obiektów danej klasy
- Tam gdzie to jest możliwe i/lub wskazane, należy użyć modyfikatora *const* i ew. listy inicjalizacyjnej.
- Tam, gdzie jest to pożądane, metody należy implementować w postaci przeciążania operatorów np. + dla klasy zbiorów (wykonującą sumowanie mnogościowe) lub + dla klasy macierzy (wykonującą dodawanie macierzy liczbowych).
- Implementację krótkich metod w postaci *inline*
- Implementację w postaci szablonu np. dla klas kontenerów.
- Obsługa błędów.

Inne

- Program powinien być - w miarę możliwości - optymalny pamięciowo i czasowo.
- Z powyższego wynika, że pamięć na dane powinna być przydzielana (i zwalniana) dynamicznie a program powinien umożliwiać pracę na danych o dowolnej długości.

Dodatkowe wymogi

- W ramach projekty oprócz kodu piszemy także **sprawozdanie**, dokumentujące zaimplementowaną klasę (lub klasy w przypadku np. dziedziczenia i/lub iteratora). W sprawozdaniu należy rozdzielić specyfikację zewnętrzną i wewnętrzną.
- Oprócz samej klasy, będącej głównym celem projektu, piszemy także **program testujący** w sposób reprezentatywny wszystkie możliwości klas(y).

TEMAT PROJEKTU 1 A

Klasa implementująca **stos**, z następującymi metodami:
- Połóż element na stosie
- Zdejmij element ze stosu
- Wypisz zawartość stosu (odpowiedni operator biblioteki *iostream*)
- Informacja, ile elementów stos zawiera.

Inne:

-Zdefiniować odpowiedni typ wyliczeniowy, informujący o stanie stosu i poprawności wykonania operacji na nim
- Jako **strukturę danych** przechowujących zawartość kontenera użyć **listę 1-kierunkową**.

Uwagi organizacyjne

- Użycie **jako struktury danych tablicy dynamicznej** (tzn. po jej przepełnieniu przydzielamy dynamicznie tablicę o 2x większym rozmiarze, po czym kopiujemy do niej wszystkie elementy) ogranicza maksymalną możliwą ilość punktów do zdobycia do 22.
- Użycie **jako struktury danych tablicy o stałym rozmiarze** ogranicza maksymalną możliwą ilość punktów do zdobycia do minimum zaliczenia, tj. 16.

TEMAT PROJEKTU 2 B

Klasa implementująca **stos**, z następującymi metodami:
- Połóż element na stosie
- Zdejmij element ze stosu
- Wypisz zawartość stosu (odpowiedni operator biblioteki *iostream*)
- Informacja, ile elementów stos zawiera.

Inne:

-Zdefiniować odpowiedni typ wyliczeniowy, informujący o stanie stosu i poprawności wykonania operacji na nim
- Jako **strukturę danych** przechowujących zawartość kontenera użyć **listę 2-kierunkową**.

Uwagi organizacyjne

- Użycie **jako struktury danych tablicy dynamicznej** (tzn. po jej przepełnieniu przydzielamy dynamicznie tablicę o 2x większym rozmiarze, po czym kopiujemy do niej wszystkie elementy) ogranicza maksymalną możliwą ilość punktów do zdobycia do 22.
- Użycie **jako struktury danych tablicy o stałym rozmiarze** ogranicza maksymalną możliwą ilość punktów do zdobycia do minimum zaliczenia, tj. 16.

TEMAT PROJEKTU 3 A

Klasa implementująca **kolejkę**, z następującymi metodami:
- Dodaj element do kolejki
- Usuń element z kolejki
- Wypisz zawartość kolejki (odpowiedni operator biblioteki *iostream*)
- Informacja, ile elementów kolejka zawiera.

Inne:

-Zdefiniować odpowiedni typ wyliczeniowy, informujący o stanie kolejki i poprawności wykonania operacji na niej
- Jako **strukturę danych** przechowujących zawartość kontenera użyć **listę 1-kierunkową**.

Uwagi organizacyjne

- Użycie **jako struktury danych tablicy dynamicznej** (tzn. po jej przepełnieniu przydzielamy dynamicznie tablicę o 2x większym rozmiarze, po czym kopiujemy do niej wszystkie elementy) ogranicza maksymalną możliwą ilość punktów do zdobycia do 22.
- Użycie **jako struktury danych tablicy o stałym rozmiarze** ogranicza maksymalną możliwą ilość punktów do zdobycia do minimum zaliczenia, tj. 16.

TEMAT PROJEKTU 4 B

Klasa implementująca **kolejkę**, z następującymi metodami:
- Dodaj element do kolejki
- Usuń element z kolejki
- Wypisz zawartość kolejki (odpowiedni operator biblioteki *iostream*)
- Informacja, ile elementów kolejka zawiera.

Inne:

-Zdefiniować odpowiedni typ wyliczeniowy, informujący o stanie kolejki i poprawności wykonania operacji na niej
- Jako **strukturę danych** przechowujących zawartość kontenera użyć **listę 2-kierunkową**.

Uwagi organizacyjne

- Użycie **jako struktury danych tablicy dynamicznej** (tzn. po jej przepełnieniu przydzielamy dynamicznie tablicę o 2x większym rozmiarze, po czym kopiujemy do niej wszystkie elementy) ogranicza maksymalną możliwą ilość punktów do zdobycia do 22.
- Użycie **jako struktury danych tablicy o stałym rozmiarze** ogranicza maksymalną możliwą ilość punktów do zdobycia do minimum zaliczenia, tj. 16.

TEMAT PROJEKTU 5 A

Klasa implementująca **zbiór** (zbiór jest kontenerem, w którym elementy nie mogą się powtarzać) z następującymi metodami:
- Dodaj/usuń element do/ze zbioru
- Funkcję opróżniającą zbiór
- Funkcję o wartościach logicznych informującą, czy dany element należy do zbioru
- Wypisz zawartość zbioru (odpowiedni operator biblioteki *iostream*)
- Informacja, ile elementów zbiór zawiera.
- Przeciążone operatory +, -, * implementujące odpowiednio operacje sumy, różnicy oraz iloczynu (części wspólnej) mnogościowych. Fakultatywnie można przewidzieć także alternatywę wykluczającą.
- Przeciążone operatory +=, -=, *=, zakodowane w postaci *inline* za pomocą wcześniej zaimplementowanych operatorów +, -, *.

Inne:

- Jako **strukturę danych** przechowujących zawartość kontenera użyć **listę 1-kierunkową**.

Uwagi organizacyjne

- Użycie **jako struktury danych tablicy dynamicznej** (tzn. po jej przepełnieniu przydzielamy dynamicznie tablicę o 2x większym rozmiarze, po czym kopiujemy do niej wszystkie elementy) ogranicza maksymalną możliwą ilość punktów do zdobycia do 22.
- Użycie **jako struktury danych tablicy o stałym rozmiarze** ogranicza maksymalną możliwą ilość punktów do zdobycia do minimum zaliczenia, tj. 16.

TEMAT PROJEKTU 6 B

Klasa implementująca **zbiór** (zbiór jest kontenerem, w którym elementy nie mogą się powtarzać) z następującymi metodami:
- Dodaj/usuń element do/ze zbioru
- Funkcję opróżniającą zbiór
- Funkcję o wartościach logicznych informującą, czy dany element należy do zbioru
- Wypisz zawartość zbioru (odpowiedni operator biblioteki *iostream*)
- Informacja, ile elementów zbiór zawiera.
- Przeciążone operatory +, -, * implementujące odpowiednio operacje sumy, różnicy oraz iloczynu (części wspólnej) mnogościowych. Fakultatywnie można przewidzieć także alternatywę wykluczającą.
- Przeciążone operatory +=, -=, *=, zakodowane w postaci *inline* za pomocą wcześniej zaimplentowanych operatorów +, -, *.

Inne:

- Jako **strukturę danych** przechowujących zawartość kontenera użyć **listę 2-kierunkową**.

Uwagi organizacyjne

- Użycie **jako struktury danych tablicy dynamicznej** (tzn. po jej przepełnieniu przydzielamy dynamicznie tablicę o 2x większym rozmiarze, po czym kopiujemy do niej wszystkie elementy) ogranicza maksymalną możliwą ilość punktów do zdobycia do 22.
- Użycie **jako struktury danych tablicy o stałym rozmiarze** ogranicza maksymalną możliwą ilość punktów do zdobycia do minimum zaliczenia, tj. 16.

TEMAT PROJEKTU 7

Klasa implementująca **listę 1-kierunkową**, wraz z zaprzyjaźnioną **klasą iteratorów.** Lista nie musi być uporządkowana. Klasa powinna umożliwiać:

- Dodawanie, kasowanie i odczytywanie elementu z początku, z końca listy i w miejscu wskazywanym przez iterator.
- Funkcję opróżniającą listę
- Funkcję o wartościach logicznych informującą, czy dany element należy do listy
- Funkcję usuwającą element(y) o danej wartości z listy (jeśli je zawiera)
- Wypisz zawartość listy (odpowiedni operator biblioteki *iostream*)
- Informacja, ile elementów lista zawiera.

TEMAT PROJEKTU 8

Klasa implementująca **listę 2-kierunkową.** Lista nie musi być uporządkowana. Fakultatywnie można zaprogramować zaprzyjaźnioną **klasę iteratorów.** Klasa powinna umożliwiać:

- Dodawanie, kasowanie i odczytywanie elementu z początku, z końca listy i w miejscu wskazywanym przez wskaźnik na element listy (ew. iterator).
- Funkcję opróżniającą listę
- Funkcję o wartościach logicznych informującą, czy dany element należy do listy
- Funkcję usuwającą element(y) o danej wartości z listy (jeśli je zawiera)
- Wypisz zawartość listy (odpowiedni operator biblioteki *iostream*)
- Informacja, ile elementów lista zawiera.

TEMAT PROJEKTU 9 A

Klasa implementująca **macierze liczbowe**. Należy pamiętać, że w matematyce numerowanie elementów macierzy zaczyna się od 1, tj. $a_{11}..a_{mn}$. Klasa powinna umożliwiać i zawierać:

- Konstruktor z parametrami umożliwiającymi określenie jej rozmiaru
- Metodę umożliwiającą dostęp do elementu macierzy, zarówno do odczytu jak i zapisu.
- Przeciążone operatory umożliwiające dodawanie, odejmowanie i mnożenie macierzy.
- Operator mnożenia powinien umożliwiać przekazania jako argument zarówno liczby jak i macierzy.

- Przeciążone operatory +=, -=, *=, zakodowane w postaci *inline* za pomocą wcześniej zaimplementowanych operatorów +, -, *.

Uwaga: Jako **strukturę danych** zapamiętującą elementy macierzy przyjąć **podwójny wskaźnik** do typu liczbowego.

W pełni optymalna struktura danych:
- Powinna umożliwiać swobodne określenie ilości wierszy i kolumn macierzy
- Elementy macierzy powinny zostać zapamiętane w spójnym obszarze pamięci (także poszczególne wiersze nie powinny być zaalokowane w różnych, rozłącznych obszarach pamięci)
- Sposób dostępu do elementów macierzy powinien być tożsamy z dostępem do predefiniowanego w C/C++ typu macierzy (o statycznie określonych rozmiarach), tj. postaci a[i][j]. Dotyczy to dostępu w implementacji metod klasy, dla użytkownika zalecane jest udostępnienie operatora postaci odpowiednio a(i,j).
- Implementacja dostępu do elementu a[1][1] (i kolejnych) nie powinna wymagać dodatkowego dodawania/odejmowania postaci +/-1.
- Uzyskanie dostępu bez powyższej wady nie powinno odbyć się kosztem utraty pamięci na nieużywane „zerowe" wiersz i kolumnę.

TEMAT PROJEKTU 10 B

Dwie klasy, łącznie implementujące **macierze liczbowe, z zastosowaniem dziedziczenia**. Należy pamiętać, że w matematyce numerowanie elementów macierzy zaczyna się od 1, tj. $a_{11}..a_{mn}$.

Uwaga: Jako **strukturę danych** zapamiętującą elementy macierzy zastosować odrębną klasę tab2D, odpowiedzialną tylko za przechowywanie danych (bez określania możliwych działań na nich). Klasa ta tab2D będzie zawierała następujące prywatne pola danych:
- zmienne całkowite określające rozmiar tablicy 2D
- **pojedynczy wskaźnik** do typu liczbowego

Ponadto klasa tab2D powinna zawierać:
- Konstruktor z parametrami umożliwiającymi określenie jej rozmiaru
- Metodę umożliwiającą dostęp do elementu macierzy, zarówno do odczytu jak i zapisu, przy czym dostęp do elementu tablicy 2D będzie wymagał mnożenia całkowitego.

Właściwa macierz będzie klasą **dziedziczącą** po klasie tab2D, rozszerzającą ja o odpowiednie działania matematyczne:

- Transponowanie
- Przeciążone operatory umożliwiające dodawanie, odejmowanie i mnożenie macierzy.
- Operator mnożenia powinien umożliwiać przekazania jako argument zarówno liczby jak i macierzy.
- Przeciążone operatory +=, -=, *=, zakodowane w postaci *inline* za pomocą wcześniej zaimplentowanych operatorów +, -, *.

TEMAT PROJEKTU 11 A

Klasa implementująca **wykres w trybie tekstowym dla funkcji zadanej w postaci wzoru w funkcji programu**. Klasa powinna umożliwiać:
- Określenie i późniejszą zmianę przedziału osi argumentów, w którym należy narysować wykres
- Automatyczne skalowanie osi y (w tym celu konieczne jest znalezienie wartości minimalnej i maksymalnej wykresu).
- Wykres powinien być zapamiętany w postaci 2-wymiarowej tablicy znakowej, klasa powinna umożliwiać jego narysowanie operatorem biblioteki *iostream*.
- Zmianę rozdzielności (tekstowej), w której ma być narysowany wykres.
- Przekazania funkcji, dla której rysujemy wykres, w postaci wskaźnika do niej.

W pełni optymalna struktura danych zapamiętująca uzyskany wykres:
- Powinna umożliwiać swobodne określenie ilości wierszy i kolumn macierzy
- Elementy macierzy powinny zostać zapamiętane w spójnym obszarze pamięci (także poszczególne wiersze nie powinny być zaalokowane w różnych, rozłącznych obszarach pamięci)
- Sposób dostępu do elementów macierzy powinien być tożsamy z dostępem do predefiniowanego w C/C++ typu macierzy (o statycznie określonych rozmiarach), tj. postaci a[i][j]. Dotyczy to dostępu w implementacji metod klasy, dla użytkownika zalecane jest udostępnienie operatora postaci odpowiednio a(i,j).
- Implementacja dostępu do elementu a[1][1] (i kolejnych) nie powinna wymagać dodatkowego dodawania/odejmowania postaci +/-1.
- Uzyskanie dostępu bez powyższej wady nie powinno odbyć się kosztem utraty pamięci na nieużywane „zerowe": wiersz i kolumnę.

TEMAT PROJEKTU 12 B

Klasa implementująca **wykres w trybie tekstowym dla funkcji zadanej w postaci wzoru w funkcji programu**. Klasa powinna umożliwiać:
- Określenie i późniejszą zmianę przedziału osi argumentów, w którym należy narysować wykres
- Automatyczne skalowanie osi y (w tym celu konieczne jest znalezienie wartości minimalnej i maksymalnej wykresu).
- Zmianę rozdzielności (tekstowej), w której ma być narysowany wykres.
- Przekazania funkcji, dla której rysujemy wykres, w postaci wskaźnika do niej.

Uwaga: Jako **strukturę danych** zapamiętującą elementy wykresu zastosować odrębną klasę tab2D, odpowiedzialną tylko za przechowywanie danych. Klasa ta tab2D będzie zawierała następujące prywatne pola danych:
- zmienne całkowite określające rozmiar tablicy 2D
- pojedynczy wskaźnik do typu znakowego

Ponadto klasa tab2D powinna zawierać:
- Konstruktor z parametrami umożliwiającymi określenie jej rozmiaru
- Metodę umożliwiającą dostęp do elementu macierzy, zarówno do odczytu jak i zapisu, przy czym dostęp do elementu tablicy 2D będzie wymagał mnożenia całkowitego.

Klasa implementująca wykres powinna dziedziczyć po klasie tab2D, rozszerzając ją o funkcjonalności właściwe dla wykresu.

TEMAT PROJEKTU 13

Klasa implementująca **macierze liczbowe z uwzględnieniem możliwości konstrukcji macierzy blokowych**, polegającą na możliwości budowy macierzy z mniejszych macierzy składowych, tej samej klasy. Klasa powinna umożliwiać i zawierać:
- Konstruktor z parametrami umożliwiającymi określenie rozmiaru
- Metodę umożliwiającą dostęp do elementu macierzy, zarówno do odczytu jak i zapisu.
- Przeciążone operatory umożliwiające dodawanie, odejmowanie i mnożenie macierzy.
- Operator mnożenia powinien umożliwiać przekazania jako argument zarówno liczby jak i macierzy.
- Operatory konstrukcji macierzy blokowych, na podstawie danych (mniejszych) macierzy składowych. Operator | odpowiada ustawieniu macierzy blokowych obok siebie, a / jedna pod drugą. Wymagana jest zgodność odpowiednich wymiarów.
- Przeciążone operatory +=, -=, *=, |=, /=zakodowane w postaci *inline* za pomocą wcześniej zaimplementowanych operatorów +, -, *.

Uwaga: Jako **strukturę danych** zapamiętującą elementy macierzy przyjąć **pojedynczy wskaźnik** do typu liczbowego.

TEMAT PROJEKTU 14

Klasa implementująca **1-wymiarowe wektory liczbowe**. Klasa powinna umożliwiać i zawierać:
- Konstruktor z parametrem umożliwiającym określenie rozmiaru
- Metodę umożliwiającą dostęp do elementu wektora, zarówno do odczytu jak i zapisu.
- Przeciążone operatory umożliwiające dodawanie, odejmowanie i iloczyn skalarny wektorów.
- Operator mnożenia powinien umożliwiać przekazania jako argument zarówno liczby jak i wektora.
- Przeciążone operatory +=, -=, *=, zakodowane w postaci *inline* za pomocą wcześniej zaimplementowanych operatorów +, -, *.
- Metody obliczania normy (suma kwadratów elementów), średniej i ekstremów (max, min).
- Sortowanie wektora.
- Wyszukiwanie elementu liniowe i połówkowe.

TEMAT PROJEKTU 15 A

Napisać obiektowo **bibliotekę rozwiązywania układów równań liniowych**. Należy zaimplementować po jednej metodzie dokładnej i iteracyjnej: Gaussa lub Gaussa-Jordana i iteracji prostej (Jacobiego). Biblioteka ma operować na obiektach klasy macierz które tworzone są dynamicznie w tablicy 2D.

TEMAT PROJEKTU 16 B

Napisać obiektowo **bibliotekę rozwiązywania układów równań liniowych**. Należy zaimplementować po jednej metodzie dokładnej i iteracyjnej: LU oraz iteracji złożonej (Seidela). Biblioteka ma operować na obiektach klasy macierz które tworzone są dynamicznie w tablicy 1D. Umożliwić wygodny dostęp do elementów macierzy poprzez operator (), wewnątrz którego dostęp do elementów odbywa się poprzez odpowiednie mnożenie.

TEMAT PROJEKTU 17 A

Napisać obiektowo program pobierający z pliku tekstowego ciąg liczb oddzielonych spacjami, na których każdej z osobna zostanie wykonana każda z funkcji, które podamy tekstowo jako argumenty linii poleceń. Wynik zapisać w pliku wyjściowym z przecinkiem jako separatorem. Zaimplementować dostępne w C++ funkcje elementarne. Na wejściu mogą się znaleźć liczby typu int oraz double.

Np. dla ciągu „1.0 2.0 3” i funkcji „sin cos” wyjście to wartości odpowiedniego typu „sin(1.0),cos(1.0),sin(2.0),cos(2.0),sin(3),cos(3)” (po wykonaniu działań).

TEMAT PROJEKTU 18 B

Napisać obiektowo program pobierający z pliku tekstowego ciąg liczb oddzielonych spacjami, na których każdej z osobna zostanie wykonane złożenie funkcji, które podamy tekstowo jako argumenty linii poleceń. Wynik zapisać w pliku wyjściowym. Zaimplementować dostępne w C++ funkcje elementarne. Na wejściu mogą się znaleźć liczby typu int oraz double.

Np. dla ciągu „1.0 3” i funkcji „sin cos” wyjście to „sin(cos(1.0)),sin(cos(3))” (po wykonaniu działań).

TEMAT PROJEKTU 19

Napisać **bibliotekę sortującą** obiekty dowolnej klasy wg. określonego pola typu np. double lub string lub innego dla którego programista zdefiniuje odpowiedni operator. Algorytmy to qsort lub merge sort, shella i proste wstawianie lub wybieranie. Dodać metody dla mierzenia czasu sortowania. Przeprowadzić porównanie dla min. jednego zestawu danych.

TEMAT PROJEKTU 20 A

Napisać obiektowo do **rekurencyjnego mnożenia macierzy**. Zaimplementować metody: z definicji oraz Strassen'a. Dodać metody dla mierzenia czasu sortowania. Przeprowadzić porównanie dla min. jednego zestawu danych. Program ma operować na obiektach klasy macierz które tworzone są dynamicznie w tablicy 2D.

TEMAT PROJEKTU 21 B

Napisać obiektowo do **rekurencyjnego mnożenia macierzy**. Zaimplementować metody: z definicji oraz Strassen'a. Dodać metody dla mierzenia czasu sortowania. Przeprowadzić porównanie dla min. jednego zestawu danych. Program ma operować na obiektach klasy macierz które tworzone są dynamicznie w tablicy 1D. Umożliwić wygodny dostęp do elementów macierzy poprzez operator (), wewnątrz którego dostęp do elementów odbywa się poprzez odpowiednie mnożenie.

TEMAT PROJEKTU 22

Napisać w oparciu o szablon (template) bibliotekę do **znajdywania w tablicy liczbowej wartości ekstremalnych** oraz ich położenia.

Zastosować algorytmy:

- pojedynczego przeglądania kolejnych elementów tablicy,
- przeglądania parami.

Każdy z powyższych dwóch algorytmów zaimplementować w wersji:

- iteracyjnej
- rekurencyjnej.

Dodać metody dla mierzenia czasu wyszukiwania. Przeprowadzić porównanie dla min. jednego zestawu danych.

TEMAT PROJEKTU 23

Napisać bibliotekę służącą do **wyszukiwania danego elementu w tablicy liczbowej**. Jako strukturę danych zastosować tablicę: nieposortowaną oraz posortowaną. Jako algorytmy wyszukiwania zastosować wyszukiwanie liniowe oraz połówkowe.

Uwzględnić wyszukiwanie połówkowe w przypadku, gdy nie jest znany przedział (lewa i prawa granica), w której znajduje się pożądana wartość. W tym celu należy dodać etap wstępny, który przy rosnących dwukrotnie w każdym kroku przyrostach, wstępnie ustali nieznany zakres indeksów.

Zastosować jako metodę prywatną dowolny algorytm sortowania. Dodać metody dla mierzenia czasu wyszukiwania. Przeprowadzić porównanie dla min. jednego zestawu danych.

TEMAT PROJEKTU 24

Napisać obiektowo program do umieszczania i wyszukiwania liczb w tablicy **metodą funkcji mieszających**. Zaimplementować min. po 3 funkcje mieszające oraz metody rozwiązywania kolizji. Dodać metody dla mierzenia czasu wyszukiwania oraz wstawiania, oraz ilości iteracji, które były niezbędne to przeprowadzenia tychże dwóch operacji. Przeprowadzić porównanie dla min. jednego zestawu danych.

TEMAT PROJEKTU 25 A

Napisać obiektowo program wyznaczający **dwuwymiarową tablicę największych wspólnych dzielników** (NWD) i najmniejszych wspólnych wielokrotności (NWW). Program ma operować na obiektach klasy macierz które tworzone są dynamicznie w tablicy 2D.

TEMAT PROJEKTU 26 B

Napisać obiektowo program wyznaczający **dwuwymiarową tablicę największych wspólnych dzielników** (NWD) i najmniejszych wspólnych wielokrotności (NWW). Program ma operowac na obiektach klasy macierz które tworzone są dynamicznie w tablicy 1D. Umożliwić wygodny dostęp do elementów macierzy poprzez operator (), wewnątrz którego dostęp do elementów odbywa się poprzez odpowiednie mnożenie.

TEMAT PROJEKTU 27

Napisać obiektowo program rozwiązujące **liniowe równania rekurencyjne** o stałych współczynnikach. Jako dane wejściowe przyjąć tablicę współczynników równania, tablicę wartości brzegowych oraz rząd równania. Np dla równania:

a(n)=5*a(n-1)-3*a(n-2)+a(n-3), a(0)=1, a(1)=5, a(2)=7

parametry wejściowe to:

tab1={5,-3,1}, tab2={1,5,7}, k=3.

Umożliwić wypełnianie tablicy kolejnymi wartościami rozwiązania równania, jak i udostępnić metodę do obliczania konkretnego elementu a(n).

Dla dociekliwych: metodę obliczania elementu a(n) zaimplementować optymalnie pod względem pamięciowym, konstruując w tablicy dynamicznej odpowiedni rejestr cykliczny.