Napisz klasę matrix. Macierz jest kwadratowa (n na n) gdzie n jest wielkością macierzy. Macierz przechowywana jest w zmiennej dynamicznej (na stercie). Silnik biblioteki matrix należy napisać samemu bez korzystania z wyspecjalizowanych bibliotek.

Funkcjonalność (metod) klasy:

- matrix(void); //konstruktor domyślny bez alokacji pamięci,
- matrix(int n); //konstruktor przeciążeniowy alokuje macierz o wymiarach n na n,
- matrix(int n , int* t); //konstruktor przeciążeniowy alokuje pamięć i przepisuje dane z tabeli,
- matrix(matrix& m); //konstruktor kopiujący,
- ~matrix(void); //destruktor,
- matrix& alokuj(int n); //jeśli macierz nie ma zaalokowanej pamięci to ją alokuje w wielkości n na n, jeśli macierz ma zaalokowaną pamięć to sprawdza czy rozmiar alokacji jest równy zdeklarowanemu rozmiarowi. W przypadku gdy tej pamięci jest mniej, pamięć ma zostać zwolniona i zaalokowana ponownie w żądanym rozmiarze. W przypadku gdy tej pamięci jest więcej pozostawić alokację bez zmian.
- matrix& wstaw(int x, int y, int wartosc); //wiersz, kolumna, wartość,
- int pokaz(int x, int y); //zwraca wartość elementu x, y,
- matrix& dowroc(void); //zamienia wiersze z kolumnami
- matrix& losuj(void); //wypełniamy cyframi od 0 do 9 wszystkie elementy macierzy
- matrix& losuj(int x); //wypełniamy cyframi od 0 do 9 elementy macierzy. Zmienna x określa ile cyfr będziemy losować. Następnie algorytm losuje, w które miejsca wstawi wylosowane cyfry,
- matrix& diagonalna(int* t); //po przekątnej są wpisane dane z tabeli, a pozostałe elementy są równe 0,
- matrix& diagonalna_k(int k, int* t); // po przekątnej są wpisane dane z tabeli, a pozostałe elementy są równe 0. Parametr k może oznaczać: 0 - przekątna przechodząca przez środek (czyli tak jak metoda diagonalna), cyfra dodatnia przesuwa diagonalną do góry macierzy o podaną cyfrę, cyfra ujemna przesuwa diagonalną w dół o podaną cyfrę,
- matrix& kolumna(int x, int* t); //przepisuje dane z tabeli do kolumny, którą wskazuje zmienna x,
- matrix& wiersz(int y, int* t); //przepisuje dane z tabeli do wiersza, który wskazuje zmienna x,
- matrix& przekatna(void); //uzupełnia macierz: 1-na przekątnej, 0-poza przekątną,
- matrix& pod_przekatna(void); //uzupełnia macierz: 1-pod przekątną, 0-nad przekątną i po przekątnej,
- matrix& nad_przekatna(void); //uzupełnia macierz: 1-nad przekątną, 0-pod przekątną i po przekątnej,

- matrix& szachownica(void); //uzupełnia macierz w ten sposób dla n=4:
  0101
  1010
  0101
  1010
- matrix& operator+(matrix& m); //A+B
- matrix& operator*(matrix& m); //A*B
- matrix& operator+(int a); //A+int
- matrix& operator*(int a); //A*int
- matrix& operator-(int a); //A-int
- friend matrix operator+(int a, matrix& m); //int+A
- friend matrix operator*(int a, matrix& m); //int*A
- friend matrix operator-(int a, matrix& m); //int-A
- matrix& operator++(int); //A++ wszystkie liczby powiększone o 1
- matrix& operator--(int); //A-- wszystkie liczby pomniejszone o 1
- matrix& operator+=(int a); //każdy element w macierzy powiększamy o „a”
- matrix& operator-=(int a); //każdy element w macierzy pomniejszamy o „a”
- matrix& operator*=(int a); //każdy element w macierzy mnożymy o „a”
- matrix& operator(double); //wszystkie cyfry są powiększone o część całkowitą z wpisanej cyfry
- friend ostream& operator<<(ostream& o, matrix& m); //wypisanie macierzy
- bool operator==(const matrix& m); //sprawdza, czy każdy element macierzy spełnia równość $A(n,m) = B(n,m)$
  A=|1 2| B=|1 2|
  |3 4| |3 4|
  jeśli nie, to nie możemy mówić, że macierze są równe,
- bool operator>(const matrix& m); //operator większości sprawdza, czy każdy element macierzy spełnia nierówność $A(n,m) > B(n,m)$. Jeśli tak, to możemy powiedzieć, że macierz jest większa, w przeciwnym wypadku nie możemy stwierdzić, że macierz jest większa.
- bool operator<(const matrix& m); //tak jak wyżej tylko operator mniejszości. Na marginesie macierzy możemy nie dać rady określić, że jest równa, mniejsza i większa, wtedy mówimy że jest różna

Klasa matrix musi być napisana w osobnym pliku. Funkcja main (też osobny plik) musi uruchamiać wszystkie metody celem sprawdzenia ich poprawności. Dobrym sposobem będzie wczytanie macierzy lub tabel z pliku aby nie wpisywać ich za każdym razem z klawiatury. Macierz powinna być testowana co najmniej na n=30 lub więcej. Należy zabezpieczyć program aby nie można było mnożyć różnych wielkości macierzy których matematycznie nie można pomnożyć.

Celem zadania jest zapoznanie się z GitHub Copilot. Na początku należy zalogować się do GitHuba i przesłać zeskanowaną swoją legitymacje studencką (która jest podbita na ten rok

akademicki). Po kilku dniach GitHub powinien włączyć Copilot'a za darmo. Następnie w Visual Studio lub Visual Studio Code trzeba doinstalować wtyczkę. Następnie trzeba utworzyć nowy projekt i zacząć programować. Projekt jest realizowany w dwuosobowych grupach.

W rozdziale o implementacji chcę aby pojawił się podrozdział w którym opiszecie jakie były trudności?, w czym AI sobie nie radził?, jakie popełniał błędy?, w czy sztuczna inteligencja pomogła?, może być kilka zrzutów kodu.

Dodatkowo do projektu należy dołączyć dokumentację w Latex wraz z doxygenem z zastosowaniem uwag, które były omawiane na poprzednich projektach. Projekt proszę realizować bez użycia narzędzi sztucznej inteligencji takich jak chaty np.: ChatGPT. Należy tylko korzystać z GitHub Cpilot'a. Projekt jest dwuosobowy i należy pisać go równolegle. Projekt należy zapisać za pomocą oprogramowania do kontroli wersji - Git oraz wysłać projekt na GitHuba. Program napisz w języku C++.