Zestaw 3

1. W zadaniu należy wykorzystać naszą klasę TString z ćwiczeń. Napisz program testujący czas wykonania wielokrotnie powtórzonych operacji: a) wstawiania na końcu kontenera, b) wstawiania na początku kontenera, c) wstawiania w dowolnym miejscu (nie na początku, nie na końcu) kontenera ale takim samym dla każdego z kontenerów, w przypadku std::vector<TString>, std::deque<TString>, std::list<TString>. Następnie tak samo dla usuwania pojedynczego obiektu a) na końcu, b) na początku, c) w dowolnym miejscu (nie na początku, nie na końcu). Jak pisać pomiar czasu można podejrzeć w zestawie „Lab 3". Sprawdzić powyższe dla kompilacji **bez** optymalizacji –O3 i **z** –O3. Wyniki czasowe zapisać jako komentarz blokowy /* */ na końcu głównego pliku programu, albo w osobnym załączonym pliku ASCII. Bez wyników zadanie będzie ocenione cząstkowo.

2. Napisz klasę o nazwie BigInt, która będzie przechowywać dowolnie wielką liczbę (całkowitą, a zatem może być ujemna) w postaci typu std::string oraz pozwalać za pomocą (na razie) metod składowych typu add(const BigInt&), subtract(const BigInt&) na obliczenia, zaś metoda print() const na wypisanie wartości na ekran. Konstruktor oraz operator= oczywiście powinien zostać napisany. Liczbę podajemy jako parametr na wejściu w postaci łańcucha znakowego.

3. Napisz program do mnożenia macierzy o zadanych rozmiarach. Macierz $A_{m\times n}$ oraz $B_{n\times p}$ daje macierz $C_{m\times p}$. Macierze wczytać z pliku ASCII, zapisując w najprostszej możliwej postaci wierszy i kolumn oddzielonych spacjami (np. macierz A będzie mieć m wierszy i n kolumn). Po przemnożeniu wynik zapisać do jakiegoś pliku. Więcej informacji oraz przykładowy program: https://eduinf.waw.pl/inf/alg/001_search/0074.php

4. Napisz program szyfrujący i deszyfrujący jakąś liczbę za pomocą algorytmu RSA. Chodzi tylko o zademonstrowanie, więc nie trzeba używać jakichś wyrafinowanych liczb pierwszych. Polecam przestudiowanie tego materiału: http://michalbereta.pl/do_pobrania/dydaktyka/RSA.pdf a także przykładowej aplikacji: https://eduinf.waw.pl/inf/alg/001_search/0067.php. Przy okazji tego zadania można się nauczyć kilku pożytecznych rzeczy oraz napisać parę innych pomocniczych programów (funkcji), np. jak zaproponowano na końcu podanego materiału.

5. Napisz program „kopiący" bitcoina. Proponuję zacząć od obejrzenia filmu https://www.youtube.com/watch?v=ZhnJ1bkIWWk (jest tam przykładowy kod w języku Python, ale jest to tak czytelne i proste, że z łatwością da się napisać to samo w języku C++, proszę przyjrzeć się funkcji konwersji https://en.cppreference.com/w/cpp/string/basic_string/to_string). Będziemy potrzebować również SHA265 i tutaj proponuję użyć gotowy i przenośny nagłówek z odpowiednią klasą (do pobrania tu: https://create.stephan-brumme.com/hash-library/). Następnie proszę w programie dodać punkty pomiaru czasu początkowego i końcowego (pomoc: dodane slajdy i materiał filmowy) i zwiększając trudność obliczeń, wypisywać czas, jaki był potrzebny na „zgadnięcie" wyniku. Odnośnik do najnowszych wartości hash BTC: https://www.blockchain.com/btc/blocks P.s. *If you are not a dumb programmer you will obviously write a for loop* :)