Celem zadania jest napisanie biblioteki implementującej część odtwarzacza multimedialnego z funkcjonalnością tworzenia i zarządzania listami odtwarzania (ang. playlist).

Główne funkcjonalności biblioteki obejmują:
* obsługę piosenek oraz filmów,
* tworzenie list odtwarzania z możliwością dodawania oraz usuwania do nich piosenek, filmów oraz innych list odtwarzania (listy odtwarzania można zagnieżdżać),
* ustawianie kolejności odtwarzania dla listy (sekwencyjnie, losowo, nieparzyste/parzyste), a następnie odtwarzanie z zadaną kolejnością.

Wszystkie błędy biblioteki powinny być zgłaszane poprzez wyjątki. Korzeniem wyjątków bibliotecznych powinna być klasa PlayerException.

Klasa Player powinna dostarczać metody publiczne:

* openFile(file) - tworzy i zwraca utwór na podstawie pliku; utwór może być odtworzony samodzielnie albo dodany do listy odtwarzania;

* createPlaylist(name) - tworzy i zwraca listę odtwarzania o podanej nazwie wraz z ustawioną domyślną (sekwencyjną) kolejnością odtwarzania; lista może zostać odtworzona samodzielnie albo dodana do innej listy odtwarzania; cykle w listach odtwarzania nie powinny być dopuszczalne.

Klasa reprezentująca listę odtwarzania powinna dostarczać następujące metody publiczne:

* add(element) - dodaje element na koniec listy odtwarzania;

* add(element, position) - dodaje element na określonej pozycji w liście odtwarzania (pozycje są numerowane od 0);

* remove() - usuwa ostatni element z listy odtwarzania;

* remove(position) - usuwa element z określonej pozycji listy odtwarzania.

* setMode(mode) - ustawia sposób (kolejność) odtwarzania utworów; sposób odtwarzania może zostać utworzony za pomocą funkcji createSequenceMode(), createShuffleMode(seed), createOddEvenMode().

* play() - odtwarza utwory na liście zgodnie z zadaną kolejnością:
    (a) sekwencyjnie - zgodnie z kolejnością na liście odtwarzania,
    (b) losowo - zgodnie z kolejnością określoną przez std::shuffle wywołane z std::default_random_engine,
    (c) nieparzyste/parzyste - najpierw wszystkie nieparzyste, a następnie wszystkie parzyste elementy listy odtwarzania.

Klasa reprezentująca piosenkę powinna dostarczać następujące metody publiczne:

* play() - odtworzenie piosenki polega na wypisaniu na standardowe wyjście napisu "Song", wykonawcy piosenki, tytułu piosenki oraz treści; treść piosenek może zawierać tylko znaki

alfanumeryczne, białe znaki oraz następujące znaki specjalne: ,.!?':;-.

Klasa reprezentująca film powinna dostarczać następujące metody publiczne:

* play() - odtworzenie filmu polega na wypisaniu na standardowe wyjście napisu "Movie", tytułu filmu, roku produkcji oraz treści; treść filmów zakodowana jest w ROT13 i może zawierać tylko znaki alfanumeryczne, białe znaki oraz następujące znaki specjalne: ,.!?':;-.

Klasa File zawiera opis pliku, w szczególności może to być opis piosenki albo filmu.
Klasa dostarcza publiczny konstruktor na podstawie napisu.
Napis zawiera kolejne części opisujące utwór. Części te są oddzielone znakiem '|'.
Pierwsza część napisu to typ (dla piosenek to będzie 'audio', a dla filmów 'video').
Ostatnia część napisu to treść utworu (dla filmów zakodowana w ROT13).
Pośrednie części napisu opisują metadane pliku w formacie name:value (dla piosenek wymagane są metadane o nazwie 'artist' oraz 'title', dla filmów wymagane są metadane o nazwie 'title' oraz 'year', dla obu mogą występować również inne metadane).

Dodatkowo należy dostarczyć funkcje tworzące różne sposoby odtwarzania dla list odtwarzania:

* createSequenceMode() - zwraca sekwencyjny sposób odtwarzania;

* createShuffleMode(seed) - zwraca losowy sposób odtwarzania z utworzonym obiektem std::default_random_engine z zadanym ziarnem;

* createOddEvenMode() - zwraca sposób odtwarzania nieparzyste/parzyste.

= Inne wymagania =

Bardzo istotną częścią zadania jest zaprojektowanie odpowiedniej hierarchii klas oraz zależności pomiędzy klasami. W szczególności nie wszystkie klasy, jakie są wymagane w rozwiązaniu, zostały jawnie wyspecyfikowane w treści zadania.

Podczas projektowania należy się kierować zasadami SOLID oraz warto uwzględnić wzorce projektowe GoF. W szczególności należy tak zaprojektować sposoby odtwarzania list, aby było możliwe dodanie kolejnego sposobu odtwarzania bez zmiany istniejącego kodu, zgodnie z zasadą Open-Closed Principle z zasad SOLID.

= Przykład użycia =

Przykład wykorzystania biblioteki znajduje się w pliku playlist_example.cc, a wynik jego uruchomienia to:

```
=== Playing 'mishmash' (default sequence mode)
Playlist [mishmash]
Movie [Cabaret, 1972]: Displaying Cabaret
Song [Dire Straits, Money for Nothing]: Now look at them yo-yo's that's the way you do it...
Playlist [armstrong]
Song [Louis Armstrong, What a Wonderful World]: I see trees of green, red roses too...
Song [Louis Armstrong, Hello, Dolly!]: Hello, Dolly! This is Louis, Dolly
```

```
Song [Dire Straits, Money for Nothing]: Now look at them yo-yo's that's the way you do it...
=== Playing 'mishmash' (shuffle mode, seed 0 for std::default_random_engine)
Playlist [mishmash]
Playlist [armstrong]
Song [Louis Armstrong, What a Wonderful World]: I see trees of green, red roses too...
Song [Louis Armstrong, Hello, Dolly!]: Hello, Dolly! This is Louis, Dolly
Movie [Cabaret, 1972]: Displaying Cabaret
Song [Dire Straits, Money for Nothing]: Now look at them yo-yo's that's the way you do it...
Song [Dire Straits, Money for Nothing]: Now look at them yo-yo's that's the way you do it...
=== Playing 'mishmash' (removed cabaret and last direstraits, odd-even mode)
Playlist [mishmash]
Playlist [armstrong]
Song [Louis Armstrong, What a Wonderful World]: I see trees of green, red roses too...
Song [Louis Armstrong, Hello, Dolly!]: Hello, Dolly! This is Louis, Dolly
Song [Dire Straits, Money for Nothing]: Now look at them yo-yo's that's the way you do it...
=== Playing 'mishmash' (sequence mode, 'armstrong' odd-even mode)
Playlist [mishmash]
Song [Dire Straits, Money for Nothing]: Now look at them yo-yo's that's the way you do it...
Playlist [armstrong]
Song [Louis Armstrong, Hello, Dolly!]: Hello, Dolly! This is Louis, Dolly
Song [Louis Armstrong, What a Wonderful World]: I see trees of green, red roses too...
unsupported type
corrupt file
corrupt content
```

= Rozwiązanie =

Rozwiązanie musi zawierać plik nagłówkowy lib_playlist.h, dający dostęp do całego publicznego interfejsu biblioteki. Oprócz tego rozwiązanie może zawierać dodatkowe pliki nagłówkowe *.h z opcjonalnymi odpowiadającymi im plikami źródłowymi *.cc.

Rozwiązanie będzie kompilowane poleceniem

```
g++ -Wall -Wextra -O2 -std=c++17 *.cc
```

Pliki rozwiązania należy umieścić w repozytorium w katalogu

grupaN/zadanie6/ab123456+cd123456

lub

grupaN/zadanie6/ab123456+cd123456+ef123456

gdzie N jest numerem grupy, a ab123456, cd123456, ef123456 są identyfikatorami członków zespołu umieszczającego to rozwiązanie. Katalog z rozwiązaniem nie powinien zawierać innych plików, ale może zawierać podkatalog prywatne, gdzie można umieszczać różne pliki, np. swoje testy. Pliki umieszczone w tym podkatalogu nie będą oceniane. Nie wolno umieszczać w repozytorium plików dużych, binarnych, tymczasowych (np. *.o) ani innych zbędnych.