Biblioteka standardowa języka C++ udostępnia bardzo przydatne kontenery (np. unordered_map i unordered_set), których nie ma w bibliotece C. Często też potrzebujemy łączyć kod w C++ z kodem w C.

Celem tego zadania jest napisanie w C++ modułu obsługującego zbiory częściowo uporządkowane (ang. poset), których elementami są napisy. Moduł ma być używany w C i ma udostępniać następujące funkcje:

```
unsigned long poset_new(void);
```

Tworzy nowy poset i zwraca jego identyfikator.

```
void poset_delete(unsigned long id);
```

Jeżeli istnieje poset o identyfikatorze id, usuwa go, a w przeciwnym przypadku nic nie robi.

```
size_t poset_size(unsigned long id);
```

Jeżeli istnieje poset o identyfikatorze id, to wynikiem jest liczba jego elementów, a w przeciwnym przypadku 0.

```
bool poset_insert(unsigned long id, char const *value);
```

Jeżeli istnieje poset o identyfikatorze id i element value nie należy do tego zbioru, to dodaje element do zbioru, a w przeciwnym przypadku nic nie robi. Nowy element nie jest w relacji z żadnym elementem. Wynikiem jest true, gdy element został dodany, a false w przeciwnym przypadku.

```
bool poset_remove(unsigned long id, char const *value);
```

Jeżeli istnieje poset o identyfikatorze id i element value należy do tego zbioru, to usuwa element ze zbioru oraz usuwa wszystkie relacje tego elementu, a w przeciwnym przypadku nic nie robi. Wynikiem jest true, gdy element został usunięty, a false w przeciwnym przypadku.

```
bool poset_add(unsigned long id, char const *value1, char const *value2);
```

Jeżeli istnieje poset o identyfikatorze id oraz elementy value1 i value2 należą do tego zbioru i nie są w relacji, to rozszerza relację w taki

sposób, aby element value1 poprzedzał element value2 (domyka relację przechodnio), a w przeciwnym przypadku nic nie robi. Wynikiem jest true, gdy relacja została rozszerzona, a false w przeciwnym przypadku.

```
bool poset_del(unsigned long id, char const *value1, char const *value2);
```

Jeżeli istnieje poset o identyfikatorze id, elementy value1 i value2 należą do tego zbioru, element value1 poprzedza element value2 oraz usunięcie relacji między elementami value1 i value2 nie zaburzy warunków bycia częściowym porządkiem, to usuwa relację między tymi elementami, a w przeciwnym przypadku nic nie robi. Wynikiem jest true, gdy relacja została zmieniona, a false w przeciwnym przypadku.

```
bool poset_test(unsigned long id, char const *value1, char const *value2);
```

Jeżeli istnieje poset o identyfikatorze id, elementy value1 i value2 należą do tego zbioru oraz element value1 poprzedza element value2, to wynikiem jest true, a w przeciwnym przypadku false.

```
void poset_clear(unsigned long id);
```

Jeżeli istnieje poset o identyfikatorze id, usuwa wszystkie jego elementy oraz relacje między nimi, a w przeciwnym przypadku nic nie robi.

Należy ukryć przed światem zewnętrznym wszystkie zmienne globalne i funkcje pomocnicze nie należące do interfejsu modułu.

Funkcje powinny wypisywać na standardowy strumień błędów informacje diagnostyczne. Poprawność wykonania funkcji, zachowanie niezmienników, spójność
danych itp. można sprawdzać za pomocą asercji. Kompilowanie z parametrem -DNDEBUG powinno wyłączać wypisywanie i asercje. Obsługa standardowego wyjścia
diagnostycznego powinna być realizowana z użyciem strumienia C++ (tzn. iostream).

Parametr value o wartości NULL jest niepoprawny.

Oczekiwane rozwiązanie powinno korzystać z kontenerów i metod udostępnianych

przez standardową bibliotekę C++. Nie należy definiować własnych struktur lub klas. Nie należy przechowywać przekazanych przez użytkownika wskaźników char const * bezpośrednio, bowiem użytkownik może po wykonaniu operacji modyfikować dane pod uprzednio przekazanym wskaźnikiem lub zwolnić pamięć. Dla każdego posetu należy przechowywać tylko jedną kopię nazwy każdego elementu.

W rozwiązaniu nie należy nadużywać kompilacji warunkowej. Fragmenty tekstu źródłowego realizujące wyspecyfikowane operacje na zbiorach nie powinny zależeć od sposobu kompilowania – parametr -DNDEBUG lub jego brak (inaczej wersja diagnostyczna nie miałaby sensu).

Przykład użycia znajduje się w pliku poset_example1.c. Przykład informacji diagnostycznych wypisywanych przez ten program znajduje się w pliku poset_example1.err.

Aby umożliwić używanie wyżej opisanych funkcji w języku C++, przy kompilowaniu
w C++ pliku nagłówkowego modułu deklaracje funkcji powinny znaleźć się w przestrzeni nazw jnp1. Przykład użycia znajduje się w pliku poset_example2.cc. Przykład informacji diagnostycznych wypisywanych przez ten program znajduje się w pliku poset_example2.err.

Przykłady można skompilować za pomocą poleceń:

```
g++ -Wall -Wextra -O2 -std=c++17 -c poset.cc -o poset.o
gcc -Wall -Wextra -O2 -std=c11 -c poset_example1.c -o poset_example1.o
g++ -Wall -Wextra -O2 -std=c++17 -c poset_example2.cc -o poset_example2.o
g++ poset_example1.o poset.o -o poset_example1
g++ poset_example2.o poset.o -o poset_example2
```

Rozwiązanie powinno zawierać pliki poset.h, poset.cc, które należy umieścić w repozytorium w katalogu

grupaN/zadanie2/ab123456+cd123456

lub

grupaN/zadanie2/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.