Instytut Informatyki i Matematyki Komputerowej UJ
Programowanie 2, semestr zimowy 2017/2018

Zadanie A
Operacje zbiorowe

Punktów do uzyskania: **10**

Język programowania: C++

autor zadania: Rafał Kawa

Generalia

- Zadanie polega na implementacji podprogramów obsługujących operacje na zbiorach obejmujących pięcioelementowe ciągi znaków `0` lub `1` nazywanych dalej prosto *zbiorami*.
- Kod rozwiązania nie może stosować:
  - Znaków kwadratowych nawiasów i ich równoważników.
  - Słów kluczowych pętli, czyli słów **`for`**, **`while`** oraz **`goto`**.
  - Rekordów, czyli słów kluczowych **`struct`** oraz **`class`**.
  - Znaków operatorów `+` oraz `-`.
  - Poleceń obsługi pamięci dynamicznej.
  - Typów własnych zmiennych innych niż **`int`**.
  - Własnych identyfikatorów zaczynających się od znaku podkreślenia.
  - Kontenerów i ogólnie szablonów.
- W wybranych podprogramach stosowany będzie porządek zbiorów w oparciu o reguły:
  - Zbiór o większej liczności jest zawsze większy od zbioru o mniejszej liczności.
  - Dla zbiorów o równej liczności zbiór poprzedzający w odwrotnej kolejności leksykograficznej elementów jest większy od zbioru następującego.

Podprogramy wymaganie w rozwiązaniu

- **`void`** `Emplace ( string, int* );`
  - Na podstawie spójnych pięcioelementowych sekwencji znaków `0` lub `1` zawartych w ciągu znakowym przekazywanym pierwszym argumentem wyznaczany jest według własnej implementacji zbiór, z adresem określenia zbioru przekazywanym drugim argumentem.
  - Ciąg znakowy przekazywany pierwszym argumentem jest dowolnie długi, obejmuje wyłącznie znaki `0`, znaki `1` lub znaki spacji, ale znaki `0` oraz `1` zawsze występują w pięcioelementowych spójnych sekwencjach.
- **`void`** `Insert ( string, int* );`
  - Pięcioelementowe spójne sekwencje znaków `0` lub `1` zawarte w ciągu znakowym przekazywanym pierwszym argumentem określają elementy dodawane do własnej implementacji zbioru, którego adres określenia jest przekazywany drugim argumentem.
  - Warunki ciągu znakowego są identyczne jak dla procedury `Emplace`.
- **`void`** `Erase ( string, int* );`
  - Pięcioelementowe spójne sekwencje znaków `0` lub `1` zawarte w ciągu znakowym przekazywanym pierwszym argumentem wyznaczają elementy usuwane z własnej implementacji zbioru o adresie określenia przekazywanym drugim argumentem.
  - Warunki ciągu znakowego są identyczne jak w procedurach `Emplace` oraz `Insert`.
- **`void`** `Print ( int, string* s );`
  - Zawartość zbioru określanego pierwszym argumentem jest przekazywana do ciągu znakowego o adresie danym drugim argumentem.
  - Elementy zbioru muszą być opisane pięcioelementową sekwencją znaków `0` lub `1` z następującą spacją.
  - Kolejnością wypisywanych elementów jest malejąca kolejność leksykograficzna elementów zbioru.
  - Zbiór pusty jest opisany słowem `empty`.
- **`bool`** `Emptiness ( int );`
  Wartościuje logicznie pustość zbioru określonego argumentem.
- **`bool`** `Nonempty ( int );`
  Wartościuje logicznie niepustość zbioru określonego argumentem.
- **`bool`** `Member ( string, int );`
  - Wartościuje logicznie zawieranie elementu określonego pojedynczą spójną pięcioelementową sekwencja znaków `0` lub `1` zawartą w ciągu znakowym przekazywanym pierwszym argumentem w zbiorze określonym drugim argumentem.
  - Ciąg znakowy oprócz dowolnej ilości spacji, zawiera dokładnie jedną pięcioelementową sekwencję znaków `0` lub `1`.
- **`bool`** `Disjoint ( int, int );`
  Wartościuje logicznie rozłączność zbiorów określanych argumentami.
- **`bool`** `Conjunctive ( int, int );`
  Wartościuje logicznie niepustość przecięcia zbiorów określonych argumentami.
- **`bool`** `Equality ( int, int );`
  Wartościuje logicznie równość zbiorów określanych argumentami.
- **`bool`** `Inclusion ( int, int );`
  Wartościuje logicznie zawieranie zbioru określonego pierwszym argumentem w zbiorze określonym drugim argumentem.
- **`void`** `Union ( int, int, int* );`
  Sumę mnogościową zbiorów określonych dwoma pierwszymi argumentami przekazuje do zbioru o adresie określenia danym trzecim argumentem.
- **`void`** `Intersection ( int, int, int* );`
  Iloczyn mnogościowy zbiorów określonych dwoma pierwszymi argumentami przekazuje do zbioru o adresie określenia danym trzecim argumentem.
- **`void`** `Symmetric ( int, int, int* );`
  Różnicę symetryczną zbiorów określonych dwoma pierwszymi argumentami przekazuje do zbioru o adresie określenia danym trzecim argumentem.
- **`void`** `Difference ( int, int, int* );`
  Różnicę mnogościową zbioru określonego pierwszym argumentem i zbioru określonego drugim argumentem przekazuje do zbioru o adresie określenia danym trzecim argumentem.
- **`void`** `Complement ( int, int* );`
  Dopełnienie mnogościowe zbioru określonego pierwszym argumentem przekazuje do zbioru o adresie określenia danym drugim argumentem.
- **`bool`** `LessThen ( int, int );`
  Wartościuje logicznie silną mniejszość zbioru określonego pierwszym argumentem względem zbioru określonego drugim argumentem.
- **`bool`** `LessEqual ( int, int );`
  Wartościuje logicznie słabą mniejszość zbioru określonego pierwszym argumentem względem zbioru określonego drugim argumentem.
- **`bool`** `GreatEqual ( int, int );`
  Wartościuje logicznie słabą większość zbioru określonego pierwszym argumentem względem zbioru określonego drugim argumentem.
- **`bool`** `GreatThen ( int, int );`
  Wartościuje logicznie silną większość zbioru określonego pierwszym argumentem względem zbioru określonego drugim argumentem.

Uzupełnienia

- W rozwiązaniu zabronione jest włączanie jakichkolwiek plików, jednakże można założyć dostępność plików nagłówkowych `iostream`, `string` oraz `sstream`.
- Jedynymi podprogramami dopuszczającymi użycie typu `string` są podprogramy `Emplace`, `Insert`, `Erase`, `Print` oraz `Member`.
- Jedynymi dopuszczalnymi metodami zmiennych typu `string` są metody `at` oraz `length`.
- Jedynym podprogramem dopuszczającym użycie typu `sstream` jest procedura `Print`.
- Jedyną dopuszczoną do użycia metodą zmiennych typu `sstream` jest metoda `str`.
- Rozwiązanie musi być zamieszczone w pliku o rozszerzeniu `cpp` i w pierwszej linii zawierać komentarz z imieniem i nazwiskiem autora.