Instytut Informatyki i Matematyki Komputerowej UJ
Programowanie 2

Zadanie A
Operacje zbiorowe

Punktów do uzyskania: **9**

Generalia

- Zadanie polega na implementacji zestawu podprogramów obsługi zbiorów w uniwersum obejmującym pięcioelementowe ciągi znaków `0` lub `1`.
- Zbiór jest reprezentowany pojedynczą daną typu **`int`**.
- Porządek zbiorów określają reguły:
  - Zbiór o większej liczności zawsze jest większy od zbioru o mniejszej liczności.
  - Dla zbiorów o równej liczności większy jest zbiór poprzedzający w odwrotnej kolejności leksykograficznej elementów.
- Rozwiązanie może zawierać wyłącznie kody wymaganych podprogramów z ewentualnymi własnymi podprogramami i w pierwszej linii musi zawierać komentarz z imieniem i nazwiskiem autora.
- Kod rozwiązania nie może stosować:
  - Włączania jakichkolwiek plików.
  - Znaków kwadratowych nawiasów i ich równoważników.
  - Słów kluczowych pętli, czyli słów **`for`**, **`while`** oraz **`goto`**.
  - Rekordów, czyli słów kluczowych **`struct`** oraz **`class`**.
  - Słowa `string`.
  - Wykorzystania pamięci dynamicznej.
  - Typów własnych zmiennych innych niż **`int`**.
  - Własnych identyfikatorów zaczynających się znakiem podkreślenia.
  - Kontenerów i ogólnie szablonów.

Wymagane podprogramy

- **`void`** `Emplace ( char*, int* );`
  Na podstawie przekazywanego pierwszym argumentem dowolnie długiego ciągu znakowego obejmującego wyłącznie spacje lub pięcioznakowe spójne sekwencje znaków `0` lub `1` wyznacza według własnej implementacji zbiór z odniesieniem przekazanym drugim argumentem.
- **`void`** `Insert ( char*, int* );`
  Elementy przekazane pierwszym argumentem o warunkach jak dla procedury `Emplace` wstawia do zbioru przekazanego drugim argumentem.
- **`void`** `Erase ( char*, int* );`
  Elementy przekazane pierwszym argumentem o warunkach jak dla procedur `Emplace` oraz `Insert` usuwa ze zbioru przekazanego drugim argumentem.
- **`void`** `Print ( int, char* );`
  Zawartość zbioru określanego pierwszym argumentem przekazuje do ciągu znakowego danego drugim argumentem, w postaci pięcioelementowych sekwencji znaków `0` lub `1` z następującą spacją i w malejącej kolejności leksykograficznej elementów. Długość danego ciągu znakowego jest minimalna dla poprawnego działania, zaś zbiór pusty jest opisany słowem `empty`.
- **`bool`** `Emptiness ( int );`
  Zwraca wartość logiczną pustości zbioru określonego argumentem.
- **`bool`** `Nonempty ( int );`
  Zwraca wartość logiczną niepustości zbioru określonego argumentem.
- **`bool`** `Member ( char*, int );`
  Zwraca wartość logiczną należenia elementu przekazanego pierwszym argumentem w postaci dowolnie długiego ciągu znakowego obejmującego wyłącznie spacje oraz dokładnie jedną pięcioznakową spójną sekwencję znaków `0` lub `1` w zbiorze określonym drugim argumentem.
- **`bool`** `Disjoint ( int, int );`
  Zwraca wartość logiczną rozłączności zbiorów określanych argumentami.
- **`bool`** `Conjunctive ( int, int );`
  Zwraca wartość logiczną niepustości przecięcia zbiorów określonych argumentami.
- **`bool`** `Equality ( int, int );`
  Zwraca wartość logiczną równości zbiorów określanych argumentami.
- **`bool`** `Inclusion ( int, int );`
  Zwraca wartość logiczną zawierania zbioru określonego pierwszym argumentem w zbiorze określonym drugim argumentem.
- **`void`** `Union ( int, int, int* );`
  Sumę mnogościową zbiorów określonych dwoma pierwszymi argumentami przekazuje do zbioru określonego trzecim argumentem.
- **`void`** `Intersection ( int, int, int* );`
  Iloczyn mnogościowy zbiorów określonych dwoma pierwszymi argumentami przekazuje do zbioru określonego trzecim argumentem.
- **`void`** `Symmetric ( int, int, int* );`
  Różnicę symetryczną zbiorów określonych dwoma pierwszymi argumentami przekazuje do zbioru określonego trzecim argumentem.
- **`void`** `Difference ( int, int, int* );`
  Różnicę mnogościową zbioru określonego pierwszym argumentem i zbioru określonego drugim argumentem przekazuje do zbioru określonego trzecim argumentem.
- **`void`** `Complement ( int, int* );`
  Dopełnienie mnogościowe zbioru określonego pierwszym argumentem przekazuje do zbioru określonego drugim argumentem.
- **`int`** `Cardinality ( int );`
  Zwraca moc zbioru danego argumentem.
- **`bool`** `LessThen ( int, int );`
  Zwraca wartość logiczną silnej mniejszości zbioru określonego pierwszym argumentem względem zbioru określonego drugim argumentem.
- **`bool`** `LessEqual ( int, int );`
  Zwraca wartość logiczną słabej mniejszości zbioru określonego pierwszym argumentem względem zbioru określonego drugim argumentem.
- **`bool`** `GreatEqual ( int, int );`
  Zwraca wartość logiczną słabej większości zbioru określonego pierwszym argumentem względem zbioru określonego drugim argumentem.
- **`bool`** `GreatThen ( int, int );`
  Zwraca wartość logiczną silnej większości zbioru określonego pierwszym argumentem względem zbioru określonego drugim argumentem.