Zadanie F: Wodzirej

Wprowadzenie.

Jakże często wracamy pamięcią do naszego dzieciństwa pełnego beztroskich gier i zabaw. Zapewne wielokrotnie brałeś udział w zabawie **chusteczka haftowana** Chociaż reguły zabawy na pewno są Ci znane, krótko je przypomnimy.

Pani przedszkolanka rozpoczynała zabawę ustawiając dzieci w kółeczku i przekazując chusteczkę jednemu z nich. Uczestnicy zabawy tańcząc i śpiewając znaną piosenkę przekazywały sobie chusteczkę haftowaną. Obowiązywała przy tym zasada, że dziewczynka zawsze przekazywała chusteczkę chłopczykowi i na odwrót.

Mogło się zdarzyć, że któreś dziecko na chwilę opuszczało zabawę, aby po chwili ponownie do niej dołączyć. Dziecko, mogło wyjść z kółeczka jedynie wtedy, gdy nie miało aktualnie chusteczki. W przeciwnym razie musiało wcześniej przekazać ją innej osobie.

Zakończenie zabawy ogłaszała przedszkolanka jednocześnie odbierając chusteczkę od dziecka, które ją aktualnie posiadało.

Zadanie.

W tym zadaniu prosimy Państwa o napisanie klasy Wodzirej.

Nazwy plików i sposób przesłania rozwiązania.

Całość implementacji musi się znaleźć w pliku source.cpp. Plik ten musi włączać plik nagłówkowy

```
#include "Uczestnik.h"
```

Plik **Uczestnik.h** będzie dostępny w czasie kompilacji i testowania Państwa rozwiązań, a jego zawartość to:

```
#ifndef __UCZESTNIK_H__
#define __UCZESTNIK_H__

struct Uczestnik{
  enum Plec{K,M,W};

  Uczestnik(Plec p) : plec(p) {}
  Plec plec;
};

#endif
```

Do systemu BaCa należy wysłać jedynie plik source.cpp zawierający kompletną implementację klasy Wodzirej i nie zawierający funkcji main. Państwa plik będzie włączany do programów testujących poprawność implementacji za pomocą

```
#include "source.cpp"
```

Rozwiązanie.

Klasa **Wodzirej** reprezentuje grupę osób bawiących się w chusteczkę haftowaną. Osoby biorące udział w zabawie są ustawione w okrąg, zwany dalej **kółeczkiem**, w ten sposób, że każda osoba ma jednoznacznie zdefiniowanego lewego i prawego sąsiada. W kółeczku jest wyróżniona osoba - **wodzirej**, która pełni rolę opiekuna zabawy.

Klasa **Wodzirej** powinna dostarczać następujących publicznych metod:

1. Konstruktor
2. dolacz - wersja 1
3. dolacz - wersja 2
4. rozpocznij
5. zakoncz
6. zrezygnuj - wersja 1
7. zrezygnuj - wersja 2
8. przekaz - wersja 1
9. przekaz - wersja 2
10. uczestnicy - wersja 1
11. uczestnicy - wersja 2
12. liczba
13. wodzirej

Szczegółowy opis metod:

1. Konstruktor domniemany. Utworzenie obiektu klasy **Wodzirej** odpowiada ogłoszeniu przez wodzireja możliwości dołączenia do kółeczka. Należy więc przyjąć, że:
   - na początku wodzirej jest pierwszą i jedyną osobą w kółeczku,
   - wodzirej jest w posiadaniu chusteczki,
   - zabawa nie jest rozpoczęta,
   - wodzirej nie jest ani mężczyzną, ani kobietą. Jego płeć odpowiada wartości **Uczestnik::W** typu wyliczeniowego zdefiniowanego w strukturze **Uczestnik**

2. Metoda o nazwie **dolacz** o jedynym argumencie typu **Uczestnik* osoba** i zwracanej wartości typu **unsigned int**. Funkcja ta powinna zwrócić
   - zero, jeśli: obiekt wskazywany wskaźnikiem **osoba** jest już ustawiony w kółeczku lub wskaźnik **osoba** jest równy zero lub nowa osoba nie jest ani mężczyzną ani kobietą.

•w przeciwnym razie zwracany jest unikatowy nieujemny identyfikator osoby. Każda kolejno dołączająca do kółeczka osoba otrzymuje identyfikator będący kolejną liczbą naturalną (zaczynamy od 1). Dodatkowo przyjmujemy, że wodzirej kółeczka ma zawsze identyfikator równy **0**.

Nowa osoba zajmuje w kółeczku taką pozycję, aby **wodzirej był jej lewym sąsiadem**.

3.Metoda o nazwie **dolacz** o dwóch argumentach typu **Uczestnik* osoba** oraz **unsigned int pozycja** i zwracanej wartości typu **unsigned int**. Funkcja ta powinna zwrócić

•zero, jeśli: obiekt wskazywany wskaźnikiem **osoba** jest już ustawiony w kółeczku lub wskaźnik **osoba** jest równy zero lub nowa osoba nie jest ani mężczyzną ani kobietą lub podana pozycja jest większa lub równa liczbie wszystkich uczestników zabawy licząc z wodzirejem.

•w przeciwnym razie zwracany jest unikatowy nieujemny identyfikator osoby, tak jak opisano w fukcji powyżej.

Drugi argument funkcji o nazwie **pozycja** wskazuje pozycję względem wodzireja, na którą **osoba** dołącza do kółeczka. Na przykład wartość zero oznacza, że osoba będzie lewym sąsiadem wodzireja, wartość jeden oznacza, że osoba będzie lewym sąsiadem lewego sąsiada wodzireja, itd.

4.Bezargumentowa metoda **rozpocznij** zwracająca wartość logiczną typu **bool**. Wartością funkcji jest **false** w każdym z następujących przypadków

•zabawa właśnie trwa (czyli nie została zakończona po uprzednim ostatnim rozpoczęciu),

•jedyną osobą w kółeczku jest wodzirej

•w kółeczku są same kobiety lub sami mężczyźni (nie bierzemy tutaj pod uwagę wodzireja)

W przeciwnym przypadku funkcja zwraca **true**. Ponadto wodzirej przekazuje chusteczkę swojemu **prawemu sąsiadowi** formalnie rozpoczynając zabawę.

5.Bezargumentowa metoda **zakoncz** zwracająca wartość logiczną typu **bool**. Wartością funkcji jest:

•**true**, jeśli zabawa właśnie trwa. Wtedy wodzirej przejmuje chusteczkę i formalnie ogłasza koniec zabawy. Osoby biorące udział w zabawie pozostają na swoich miejscach w kółeczku.

•**false** jeśli zabawa nigdy nie została rozpoczęta lub nie została ponownie rozpoczęta po jej ostatnim zakończeniu.

6.Metoda **zrezygnuj** o jednym argumencie typu **Uczestnik* osoba** oraz zwracanej wartości typu **bool**. Metoda zwraca **false** w każdym z następujących przypadków

•osoba wskazywana wskaźnikiem **osoba** nie stoi w kółeczku,

•osoba wskazywana wskaźnikiem **osoba** jest w posiadaniu chusteczki,

•osoba wskazywana wskaźnikiem **osoba** to wodzirej,

•wskaźnik **osoba** jest równy zero.

W przeciwnym razie funkcja zwraca **true**, a dodatkowym efektem działania funkcji jest wyjście tej osoby **osoba** z kółeczka. Prawy sąsiad osoby wychodzącej z kółeczka staje się prawym sąsiadem lewego sąsiada osoby wychodzącej. Analogicznie, lewy sąsiad osoby wychodzącej staje się lewym sąsiadem prawego

sąsiada osoby wychodzącej.
Uwaga: osoba może wyjść z kółeczka, a po pewnym czasie ponownie do niego dołączyć niekoniecznie na tą samą pozycję. Otrzymuje wtedy **nowy identyfikator**.
7.Metoda **zrezygnuj** o jednym argumencie typu **unsigned int** i zwracanej wartości typu **bool**. Tym razem, osoba wychodząca z kółeczka jest jednoznacznie określona przez jej unikatowy identyfikator (argument funkcji). Przypomnijmy, że identyfikator ten został nadany osobie w czasie dołączania do kółeczka. Metoda zwraca **false** w każdym z następujących przypadków

- •osoba o podanym identyfikatorze nie stoi w kółeczku,
- •osoba o podanym identyfikatorze jest w posiadaniu chusteczki,
- •osoba o podanym identyfikatorze to wodzirej.

W przeciwnym razie metoda zwraca **true**, a dodatkowym skutkiem jest wyjście osoby z kółeczka, na zasadach takich samych, jako opisano w pierwszej wersji funkcji **zrezygnuj**.
8.Metoda **przekaz** o jedynym argumencie typu **Uczestnik* osoba** i zwracanej wartości typu **bool**. Metoda zwraca **false** w każdym z następujących przypadków

- •zabawa nigdy nie została rozpoczęta,
- •zabawa nie została ponownie rozpoczęta po jej uprzednim zakończeniu,
- •osoba pokazywana wskaźnikiem **osoba** nie stoi w kółeczku,
- •osoba pokazywana wskaźnikiem **osoba** jest tej samej płci jak osoba, która jest w posiadaniu chusteczki.
- •osoba pokazywana wskaźnikiem **osoba** to wodzirej

W przeciwnym razie metoda zwraca **true**, a chusteczka przechodzi w posiadanie osoby wskazywanej przez **osoba**.
9.Metoda **przekaz** o jedynym argumencie typu **unsigned int** i zwracanej wartości typu **bool**. Metoda zwraca **false** w każdym z następujących przypadków

- •zabawa nigdy nie została rozpoczęta,
- •zabawa nie została ponownie rozpoczęta po jej uprzednim zakończeniu,
- •w kółeczku nie ma osoby o podanym identyfikatorze,
- •osoba o podanym unikatowym identyfikatorze jest tej samej płci jak osoba, która jest w posiadaniu chusteczki,
- •osoba o podanym unikatowym identyfikatorze to wodzirej.

W przeciwnym razie metoda zwraca **true**, a chusteczka przechodzi w posiadanie osoby o identyfikatorze podanym jako argument funkcji.
10.Stała, bezargumentowa i bez wartości (void) metoda **uczestnicy**. Jej zadaniem jest wypisanie na standardowe wyjście wszystkich osób ustawionych w kółeczku (łącznie z wodzirejem) w następujący sposób:

- •osoby są wypisywane kolejno począwszy od wodzireja, następnie lewy sąsiad wodzireja, następnie lewy sąsiad lewego sąsiada wodzireja, itd.
- •każda osoba w kółeczku jest wypisywana w oddzielnej linii w formacie (nie ma dodatkowych białych znaków):

```
plec: P, nr: N
```

gdzie **P** oznacza płeć osoby (dla **K** wartość **0**, dla **M** wartość **1**, dla **W** wartość **2**), a **N** oznacza unikatowy identyfikator osoby.

11.Stała metoda **uczestnicy** o jedynym argumencie typu **Uczestnik::Plec p** (bez wartości - void). Jej zadaniem jest wypisanie na standardowe wyjście wszystkich osób w kółeczku o płci równej **p** w następujący sposób :

•osoby są sprawdzane i wypisywane kolejno począwszy od wodzireja, następnie prawy sąsiad wodzireja, następnie prawy sąsiad prawego sąsiada wodzireja, itd.
•każdy z uczestników zabawy jest wypisywany w oddzielnej linii w formacie (nie ma dodatkowych białych znaków):

```
nr: N
```

gdzie **N** oznacza unikatowy identyfikator uczestnika,
•wypisywani są tylko uczestnicy, dla których atrybut **plec** jest równy **p**.

12.Bezargumentowa, stała metoda **liczba** zwracająca liczbę osób uczestniczących w zabawie licząc **łącznie z wodzirejem**. Typ zwracanej wartości to **unsigned int**.
13.Bezargumentowa i stała metoda **wodzirej** zwracająca stały wskaźnik do obiektu typu **Uczestnik**. Wartością funkcji jest adres jedynego wodzireja w kółeczku.

Inne uwagi.

•W nadesłanym rozwiązaniu nie wolno korzystać z kolekcji standardowych STL!
•W nadesłanym rozwiązaniu nie wolno korzystać z tablic, w szczególności nie można użyć żadnego z symboli: **[]**.
•Poprawnie napisany program powinien odpowiednio zarządzać zasobami (np. zwalniać pamięć, która została mu przydzielona).
•Klasa **Wodzirej** powinna być możliwie odporna na błędy użytkownika (w slangu informatycznym - idiotoodporna:).

Przykładowe użycie klasy.

Wyjściem następującego programu

```
#include <iostream>
#include "source.cpp"
using namespace std;

int main()
{
  Wodzirej w;

  Uczestnik p(Uczestnik::K);
  Uczestnik q(Uczestnik::M);
  Uczestnik r(Uczestnik::K);
  Uczestnik s(Uczestnik::M);
  Uczestnik t(Uczestnik::K);
  Uczestnik u(Uczestnik::M);
```

```
  cout << "####################\n";
  w.dolacz(&p);
  cout << w.dolacz(&q) << endl;
  w.dolacz(&r,0);
  w.dolacz(&s,1);
  w.dolacz(&t);
  w.dolacz(&u);
  cout << w.liczba() << endl;
  cout << "####################\n";
  w.uczestnicy();
  w.uczestnicy(Uczestnik::K);
  cout << "####################\n";
  w.zrezygnuj(2);
  w.uczestnicy();
  w.uczestnicy(Uczestnik::M);
  cout << "####################\n";

  return 0;
}
```

powinno być

```
####################
2
7
####################
plec: 2, nr: 0
plec: 0, nr: 3
plec: 1, nr: 4
plec: 0, nr: 1
plec: 1, nr: 2
plec: 0, nr: 5
plec: 1, nr: 6
nr: 5
nr: 1
nr: 3
####################
plec: 2, nr: 0
plec: 0, nr: 3
plec: 1, nr: 4
plec: 0, nr: 1
plec: 0, nr: 5
plec: 1, nr: 6
nr: 6
nr: 4
####################
```