Zadanie 1

Napisz funkcję, która przyjmie parę (szablon pair) liczb całkowitych (zakładamy, że pierwsza jest zawsze mniejsza od drugiej). Funkcja powinna zwracać wektor kolejnych, rosnących liczb całkowitych z zadanego przedziału (zaczynając od mniejszej, a kończąc na większej minus 1).

Przykładowo dla pary zawierającej wartości (10, 15) powinien zostać zwrócony wektor:
10, 11, 12, 13, 14

Zadanie 2

Napisz funkcję, która będzie przyjmowała parę (szablon pair) wektorów liczb całkowitych. Funkcja powinna sprawdzić (i zwrócić jako bool) czy zawartość obu wektorów jest identyczna. Nie ma gwarancji, że wektory będą tego samego rozmiaru.

Deklaracja pary wektorów: pair< vector<int>, vector<int> >
(spacja zaznaczona na niebiesko jest **obowiązkowa** w starszych standardach c++)

Zadanie 3

Napisz interfejs Przeciwnik, zawierający abstrakcyjną metodę **int zycie()**.

Napisz klasy Smok i Wilk, realizujące interfejs Przeciwnik. Jako liczbę punktów życia smoki powinny zwracać 7, a wilki 4.

Następnie napisz klasę Plansza. Klasa powinna zawierać (jako prywatne pole) kontener wskaźników na obiekty realizujące interfejs Przeciwnik (np. vector lub deque). W konstruktorze klasy Plansza należy wypełnić kontener kilkoma dynamicznie tworzonymi obiektami klas Smok i Wilk (liczby wilków i smoków mogą być generowane losowo, albo np. przyjmowane jako argument konstruktora).

Dodatkowo klasa Plansza powinna zawierać metodę **int poziomTrudnosci()**. Poziom trudności danej planszy powinien być obliczany jako suma punktów życia wszystkich znajdujących się na niej przeciwników. Kod powinien być napisany w taki sposób, aby ewentualne późniejsze dodanie klas nowych przeciwników nie wymagało modyfikacji tej metody.

Zadanie 4

Napisz klasę, która będzie zawierała statyczny kontener wskaźników na obiekty tej klasy.

Kiedy obiekt tej klasy będzie tworzony, konstruktor powinien dodawać odpowiedni wskaźnik do kontenera. Analogicznie, kiedy obiekt będzie usuwany, destruktor powinien usuwać odpowiedni wskaźnik z kontenera.

Konstruktor klasy powinien przyjmować identyfikator typu string, który będzie zapisywany w prywatnym, stałym polu. Klasa powinna zawierać również getter dla tego identyfikatora.

Na końcu do klasy należy dopisać **statyczną** metodę listaObiektow(). Metoda powinna wypisywać na ekran identyfikatory wszystkich istniejących obiektów tej klasy.

Uwagi:

Zadanie czwarte najprościej będzie zrealizować z użyciem kontenera **set –** metody insert() i erase() pozwolą na łatwe dodawanie i usuwanie elementów. Oczywiście można użyć innego kontenera, np. vector, ale trzeba będzie zastanowić się nad poprawną metodą usuwania wskaźnika.