

Zadanie 6

W poniższym kodzie szablon funkcji **part** deklaruje jako trzeci argument (FUN) *cokolwiek* co da się wywołać (wskaźnik funkcyjny, lambda) z argumentem typu **T** i zwraca **bool** (takie funkcje nazywamy *predykatami*).

Uzupełnij kod programu, tak, aby dał się skompilować i wykonać.

Funkcja (szablon) **printTab** ma za zadanie wydrukować w ładnej formie przekazaną tablicę (elementy w jednym wierszu, oddzielone spacjami).

Funkcja (szablon) **part** ma za zadanie tak poprzestawiać elementy przekazanej tablicy **arr**, aby wszystkie elementy, dla których predykat **FUN** jest spełniony (tzn. **FUN** wywołany z wartością tego elementu jako argumentem zwraca **true**) znalazły się na lewo od wszystkich elementów, dla których ten predykat nie jest spełniony. Jako argumentu odpowiadającego parametrowi **FUN** można użyć wskaźnika do funkcji typu **T→bool** (jak w linii 23) lub lambdy o takiej sygnaturze. W wynikowej tablicy względna kolejność elementów w ramach tych, które predykat spełniają i tych, które go nie spełniają, jest dowolna. Funkcja zwraca indeks pierwszego elementu, który *nie* spełnia predykatu; zauważ, że jest to jednocześnie liczba elementów, które predykat *spełniają* (być może 0 lub **size**).

[download FunTmpl.cpp](#)

```

1  #include <iostream>
2  #include <cstdlib>
3  #include <ctime>
4  #include <functional>
5  using namespace std;
6
7  template <typename T, typename FUN>
8  size_t part(T* arr, size_t size, FUN f) {
9      // ...
10 }
11
12 template <typename T>
13 void printTab(const T* t, size_t size) {
14     // ...
15 }
16
17 bool isEven(int e) { return e%2 == 0; }
18
19 int main() {
20
21     size_t ind = 0;
22

```

```

23     int a1[] = {1,2,3,4,5,6};
24     ind = part(a1,6,isEven);
25     cout << "ind = " << ind << " ";
26     printTab(a1,6);
27
28     int a2[] = {1,2,3,4,5,6};
29     // lambda as argument: a predicate checking
30     // if the given number is odd
31     ind = part( /* ... */ );
32     cout << "ind = " << ind << " ";
33     printTab(a2,6);
34
35     double a3[] = {-1.5,2.5,3.5,6.5,4.5,0};
36     double mn =2.0;
37     double mx =5.0;
38     // lambda as argument: a predicate checking
39     // if the given number is in the range [mn,mx]
40     ind = part( /* ... */ );
41     cout << "ind = " << ind << " ";
42     printTab(a3,6);
43
44     constexpr size_t DIM = 500000;
45     int* a4 = new int[DIM];
46     srand(unsigned(time(0)));
47     for (size_t i = 0; i < DIM; ++i) a4[i] = rand()%21+1;
48     // lambda as argument: a predicate checking
49     // if the given number is divisible by 7
50     ind = part( /* ... */ );
51     cout << "ind = " << ind << endl;
52     delete [] a4;
53 }

```

Fragment w liniach 43-46 służy do wygenerowania wartości do zainicjowania tablicy **a4** (wszystkie wartości będą pochodzić z przedziału [1,21]).

Operację rozdzielania elementów należy przeprowadzić w *jednej* pojedynczej pętli (bez pętli zagnieżdżonych), inaczej wywołanie z linii 49 dla tablicy o wymiarze pół miliona, trwałoby zbyt długo; przy poprawnej implementacji wykonanie powinno być praktycznie natychmiastowe. W funkcji **part** *nie* wolno tworzyć żadnych pomocniczych tablic!

Przykładowy wynik programu mógłby wyglądać tak:

```

ind = 3 [ 2 4 6 1 5 3 ]
ind = 3 [ 1 3 5 4 2 6 ]
ind = 3 [ 2.5 3.5 4.5 6.5 -1.5 0 ]
ind = 71461

```

Termin: do 16 maja (włącznie)

Rozwiązania, w postaci **jednego** pliku źródłowego zawierającego treść programu, proszę wrzucać w systemie EDU do katalogu „Foldery zadań / Zadanie_XX”, gdzie 'XX' jest numerem zadania.

Nazwą pliku powinno być nazwisko z dużej litery (bez polskich znaków); rozszerzeniem musi być '.cpp', czyli np. `Malinowska.cpp`.