Sprawdzian

pon
10:00 C

imię i nazwisko _____

liczba punktów _____

dokument klasy **B1**, archiwizować do **2020-01-01**

W programie zdefiniowano następujące typy:

```
1  struct Komisja
2  {
3      int         numer;
4      Kandydat * kandydaci; // wskaznik na glowe listy kandydatow
5      Komisja  * next;      // wskaznik na nastepna komisje
6  };
7
8  struct Kandydat
9  {
10     std::string imie, nazwisko;
11     int         liczba_glosow;
12     Kandydat *  next; // wskaznik na nastepnego kandydata
13 };
```

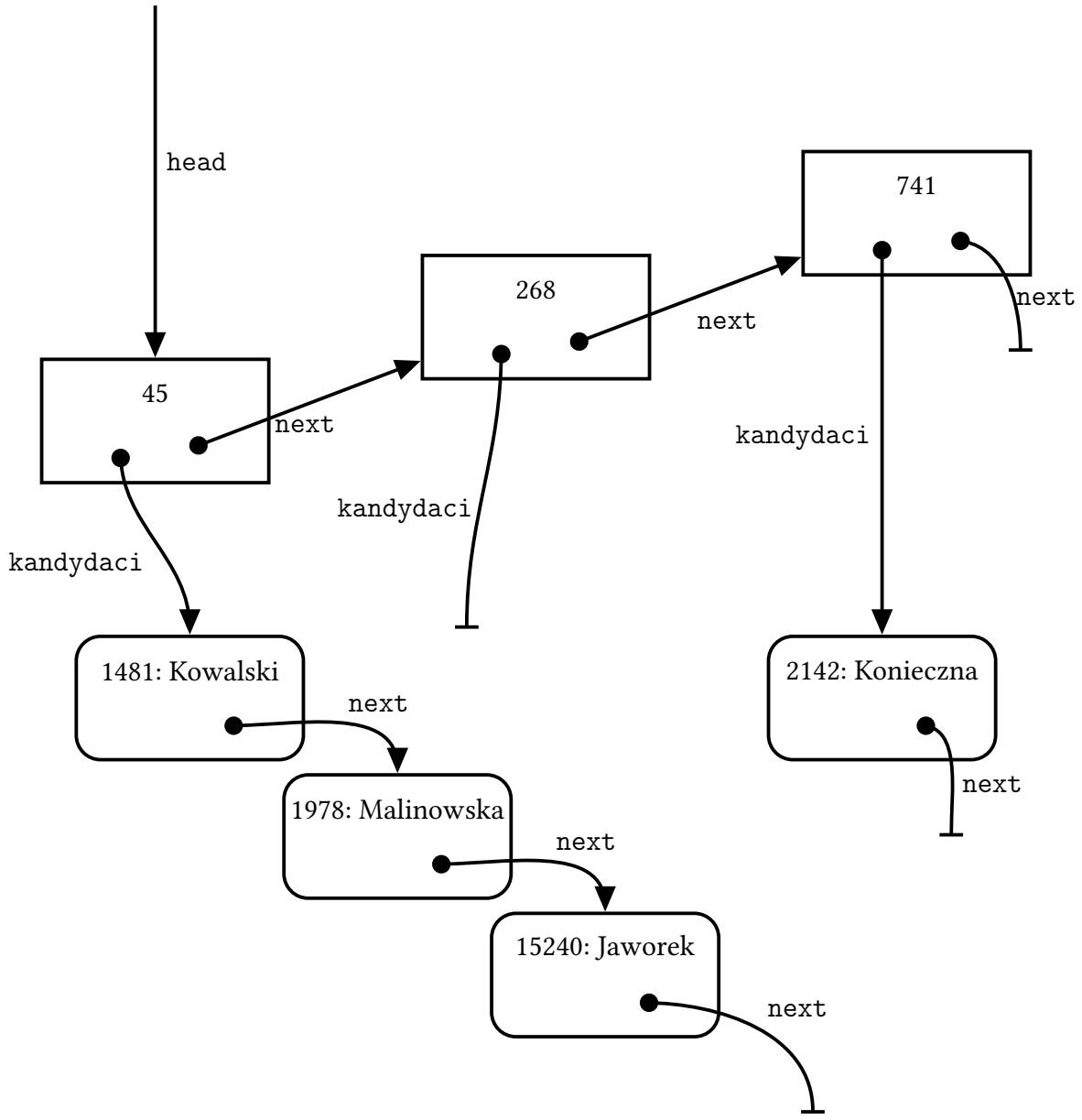
Korzystając z nich można utworzyć strukturę danych, której przykład jest przedstawiony na rysunku 1. Komisje tworzą listę jednokierunkową uporządkowaną rosnąco wg numerów. Kandydaci przypisani do komisji tworzą listę jednokierunkową uporządkowaną niemalejąco wg liczby uzyskanych głosów. Możliwe jest, że komisja nie ma jeszcze żadnego kandydata lub ma ich dowolną liczbę.

Zadanie

Proszę napisać funkcję `usun`, która usuwa kandydata z komisji. Do funkcji należy przekazać numer komisji, imię i nazwisko kandydata. Możliwe jest, że kandydata nie ma w żadnej komisji lub w ogóle nie ma żadnej komisji. Gdy po usunięciu kandydata komisja nie ma żadnego kandydata, należy usunąć komisję. Funkcję `usun` należy w sposób roztropny podzielić na podfunkcje.

Zadanie domowe

Proszę napisać funkcję `aktualizuj`, która zwiększa liczbę głosów kandydata o zadaną wartość. Do funkcji należy przekazać numer komisji, imię i nazwisko kandydata i liczbę głosów, o jaką należy zaktualizować konto kandydata. Jeżeli nie ma komisji, należy komisję dodać. Jeżeli nie ma kandydata, należy dodać kandydata. Należy zapewnić, że lista komisji i listy kandydatów są należycie uporządkowane. Szczególnie należy zwrócić uwagę, że zmiana liczby głosów może spowodować konieczność przeniesienia kandydata w liście danej komisji. Funkcję `aktualizuj` należy w sposób roztropny podzielić na podfunkcje.



Rysunek 1: Przykładowa struktura danych przechowująca komisje i przypisanych im kandydatów.