Podczas ćwiczeń będziemy stale posługiwać się następującymi oznaczeniami typów danych reprezentujących elementy prostych struktur listowych ze wskaźnikami:

```
typedef struct elem{ int x; struct elem *nast; } el;
typedef el *wel;
```

1. Napisać funkcje wykonujące podstawowe operacje na liniowej liście jednokierunkowej zbudowanej z elementów typu `el`: a) dodanie elementu o wartości `y` na początku listy, b) na końcu listy, c) wyszukiwanie elementu o zadanej wartości `x`, d) dodanie nowego elementu przed oraz e) za wskazanym elementem listy, f) usunięcie pierwszego elementu, g) usunięcie ostatniego, h) usunięcie elementu wskazanego. Funkcje powinny poprawnie działać również wtedy gdy lista, na której operują jest pusta.

2. Zadanie 1 należy rozwiązać dwoma sposobami i zaobserwować podobieństwa oraz różnice obydwu wariantów:

a) po pierwsze: parametr funkcji jest wskaźnikiem na listę, a wynik (wskaźnik na nowy początek listy) zwracany jest jako wartość funkcji, `wel dodaj_pocz(wel L, int y){...}`

b) po drugie: parametr funkcji jest *adresem* wskaźnika na listę i wykorzystywany jest do przekazania wyniku operacji, `void dod_pocz(wel *L, int y){...}`

Uwaga: W kolejnych zadaniach przyjmujemy jako standard, że funkcje, które w jakikolwiek sposób *modyfikują* zawartość lub strukturę listy, odwołują się do niej przez *adres wskaźnika*, np. `void zamień(wel *L, ...)`, natomiast te, które jedynie odczytują informacje z listy bez modyfikowania jej zawartości, odwołują się do niej przez *wartość wskaźnika*, np. `void drukuj(wel L)`.

3. Zmodyfikować rozwiązania uzyskane w punkcie b) zad. 2 w taki sposób, aby wszystkie operacje dodawania nowego elementu korzystały z wywołania funkcji tworzącej nowy element na początku listy. Podobnie, funkcje usuwające elementy z listy powinny korzystać z operacji usuwania pierwszego jej elementu.

4. Z listy wskazywanej przez zmienną `P` usunąć wszystkie elementy o wartości `y`. W wyniku tej operacji lista może stać się pusta.

5. Wydrukować zawartość listy w porządku: a) prostym; b) odwrotnym.

6. Zaprogramować operację odwracania listy (bez generowania nowych elementów).

7. Z wczytywanych danych utworzyć listę uporządkowaną: a) z powtarzającymi się danymi, jeśli takie występują; b) z eliminacją powtórzeń. Napisać dwie wersje rozwiązania — bez "wartownika" i z "wartownikiem".

8. Napisać procedurę usuwającą z listy liniowej powtarzające się elementy, zakładając że lista a) jest, b) nie jest uporządkowana.

9. Połączyć dwie uporządkowane listy wskazywane przez zmienne `P` i `Q` w jedną listę uporządkowaną. W wyniku tej operacji `P` wskazuje na połączoną listę, a `Q = NULL`.
Uwaga: Poszukujemy rozwiązania optymalnego, sprytniejszego niż przenoszenie z `Q` do `P` elementu po elemencie, tj. rozwiązania, w którym przenoszeniu podlegają — o ile to możliwe — całe uporządkowane serie danych. Przeanalizować, czy w tym wypadku użycie list z wartownikiem upraszcza istotnie kod rozwiązania.

10. Podać rekursywne rozwiązania zadań 4–9.

11. Napisać procedury dołączania i usuwania elementów z listy dwukierunkowej: na początku listy, na końcu oraz przed lub za wskazanym elementem.

12. Zrealizować repertuar elementarnych operacji na jednokierunkowej liście cyklicznej, gdzie wskaźnik na listę pokazuje jej początkowy element: dodawanie nowego elementu na początku i na końcu cyklu, usuwanie pierwszego i ostatniego elementu cyklu, odnajdywanie elementu o zadanej wartości, dodawanie nowego elementu przezd i po wskazanym, usuwanie wskazanego elementu.

13. Napisać funkcję, która usuwa powtórzenia z listy cyklicznej.

14. Napisać funkcję, która zwraca wartość elementu nieuporządkowanej listy cyklicznej o maksymalnej liczbie powtórzeń.

15. Kolejka jest strukturą listową, w której nowe elementy dodawane są zawsze na końcu, `push(&Q,x)`, natomiast podczas odczytu pobierany jest zawsze początkowy element listy, `y=pop(&Q)`. Jest on przy tym usuwany z kolejki. Zapropowować prostą realizację kolejki w postaci listy cyklicznej. Jakie są zalety takiego rozwiązania w stosunku do realizacji za pomocą listy otwartej?

16. **Projekt:** Zaproponować algorytm tworzenia skorowidza do tekstu książki. Na ćwiczeniach szczegółowo omówimy założenia algorytmu. Powinien on tworzyć dynamiczną strukturę danych, która na końcu posłuży do wydruku skorowidza w postaci zbliżonej do:

parametr 103, 112–115
prefiks 75, 99, 120
procedura 30, 45–49, 78
...