AiSD Lista nr 1 (9.X.2017)

1. Napisać funkcje wykonujące podstawowe operacje na liniowej liście jednokierunkowej przyjmując, że jej węzły mają następującą strukturę:

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

Repertuar operacji do zaimplementowania obejmuje: dodanie elementu o wartości `y` na początku listy, na końcu listy, wyszukiwanie elementu o zadanej wartości `x`, dodanie nowego elementu przed oraz za wskazanym elementem listy, usunięcie pierwszego elementu, usunięcie ostatniego, 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 i różnice obydwu wariantów:

- po pierwsze: parametr funkcji jest wskaźnikiem na listę, a wynik zwracany jest jako wartość funkcji, `el *dod_pocz(el *L, int y){...}`
- po drugie: parametr funkcji jest *adresem* wskaźnika na listę i wykorzystywany jest do przekazania wyniku operacji, `void dod_pocz(el **L, int y){...}`

3. Z listy wskazywanej przez zmienną `P` usunąć wszystkie elementy o wartości `y`. Porównać dwa warianty rozwiązania jak w zadaniu 2:

`P=usun_wsz(P,y);` oraz `usun_wsz(&P,y);`

Uwaga: w wyniku tej operacji lista może stać się pusta.

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

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

6. 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".

7. Jak poprzednio, porównać dla zadania 6 dwa warianty komunikowania się funkcji z otoczeniem.

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 3, 4, 5, 6, 8, 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: dodawanie nowego elementu przed i po elemencie wkazywanym jako "początek" cyklu, odnajdywanie elementu o zadanej wartości, 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. **Projekt:** Zaproponować algorytm tworzenia skorowidza do tekstu książki. Szczegółowe omówienie założeń algorytmu — na ćwiczeniach. Algorytm powinien 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
...