Zadania laboratoryjne do zajęć „Algorytmy i struktury danych”

1. Wczytać ciąg liczb, tworząc z nich listę przez dołączanie kolejnych elementów a) na początku, b) na końcu listy. Następnie wydrukować tę listę od pierwszego do ostatniego elementu. Zadbać o właściwe warunki zakończenia pętli.
2. Uzupełnić program z zadania 1 o rekursywne procedury drukowania listy a) w porządku prostym (od początku do końca), b) odwrotnym.
3. Napisać funkcję znajdź(P,n) o wartościach typu wskaźnikowego, która na liście wskazywanej przez P lokalizuje element o wartości n (pierwszy jeśli występują powtórzenia) i zwraca wskaźnik na ten element lub **nil** gdy n nie występuje na liście. Następnie napisać procedurę dołącz(n,P) dodającą element o wartości n do listy, bezpośrednio za elementem wskazywanym przez P. Wykorzystać te narzędzia do utworzenia listy uporządkowanej z wczytywanych danych a) z ewentualnymi powtórzeniami, b) eliminując powtarzające się elementy. Wydrukować kontrolnie wynikową listę.
4. Uzupełnić program z zad. 3 o fragment, który po utworzeniu listy czyta z wejścia pojedynczą liczbę, lokalizuje ją na liście i informuje o jej nieznalezieniu lub znalezieniu, a następnie usuwa odnośny element listy. Przetestować działanie programu dla różnych ewentualności, także gdy lista jest pusta.
5. Napisać program odwracający daną listę bez generowania nowych elementów (tj. wyłącznie przez manipulacje na łącznikach). Podać rozwiązanie a) iteracyjne b) rekursywne.
6. Korzystając z podprogramów z zad. 3. utworzyć dwie oddzielne listy uporządkowane z dwóch serii danych, a następnie napisz funkcję, która dokona połączenia dwóch list w jedną listę uporządkowaną.
7. Napisać procedury/funkcje lokalizujące, dołączające i usuwające elementy danych na liście cyklicznej. Przetestować kontrolnym wydrukiem działanie programu dla różnych ewentualności, także gdy lista jest pusta.
8. Zaimplementuj obsługę kolejki priorytetowej. Węzeł kolejki oprócz pola klucz musi posiadać pole *piorytet,* które będzie określało ważność danego elementu listy. Elementy na liście zawsze muszą być umieszczane i usuwane z zachowaniem porządku względem priorytetów.
    a. dodaj(K, w, p) - dodaje do kolejki K wartość w o priorytecie p,
    b. wez(K) - zwraca (i usuwa z kolejki K) element o najwyższym priorytecie,
    c. zmPrior(K, w, nowyPrior) - zmienia priorytet elementu(-ów) o wartości w.