Algoryty i Struktury Danych 2012

Lista 1

1. Przyjmij następującą definicję węzła:

```
struct node
{   int key;
    node* next;
    node(int k0,node* n0=NULL):key(k0),next(n0){}
};
```

   Napisz procedury:

   (a) `void insert(node*& l,int x)` – umieszcza x na początku listy l

   (b) `int pop(node*& l)` – zwraca pierwszy element listy i usuwa go z listy.

   (c) `lnode* find (lnode* l, int x)` – zwraca wskaźnik na wezęł zawierający klucz x lub NULL gdy nie ma takiego węzła.

   (d) `int sum(node* l)` – zwraca sumę elementów listy l.

   (e) `void reverse(node*& l)` – odwraca kolejność elementów listy l.

2. Dana jest tablica n elementowa. `int t[n];` Napisz funkcję `void shift(int t[],int n, int k)` która przesuwa elementy tablicy `t` cyklicznie prawo o k pozycji. Jeśli $k < 0$ to oznacza przesunięcie w lewo. Jeśli $k > n$ to przesuwa o $k \bmod n$ pozycji. Ilość dodatkowej pamięci użytej przez procedurę musi być stała (niezależna od n).

3. Zakładając, że elementy tablicy t są posortowane napisz funkcję `int gdzie(int t[], int x)`, która zwraca pozycję w tablicy t na której występuje element x. Użyj metody bisekcji. Jaka jest złożoność czasowa twojego algorytmu? Jaka byłaby średnia złożoność (wartość oczekiwana faktycznie wykonanej liczby porównań) dla tablicy nieposortowanej w przypadku gdy: (a) element x jest w tablicy (b) elementu x nie ma w tablicy t.

4. Spróbuj samodzielnie napisać procedurę `insertion_sort(int t[], int n)`.

5. Napisz procedurę `node *insertion_sort(node*& l)`, która usuwa elementy z nieposortowanej listy $l0$ i tworzy nową posortowaną listę. Wynikiem jest adres pierwszego elementu posortowanej listy.