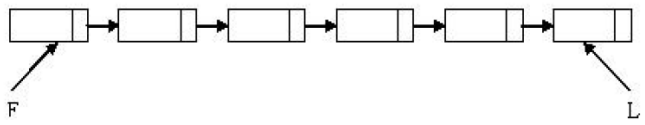
Úlohy na precvičenie na priebežný test

1. Uvažujte jednosmerne zreťazený zoznam vo verzii s dvomi ukazovateľmi F a L doňho:



Vyznačte tie operácie, ktorých čas vykonania záleží od dĺžky zoznamu a stručne uveďte, prečo:

A. Zrušenie posledného prvku zoznamu.
B. Zrušenie prvého prvku zoznamu.
C. Pridanie prvku za posledný prvok zoznamu.
D. Pridanie prvku pred prvý prvok zoznamu.
E. Vymenenie prvých dvoch prvkov zoznamu.

2. Rozptylová (hashovacia) funkcia

a) transformuje adresu daného prvku na jemu príslušný kľúč
b) vracia pre každý kľúč jedinečnú hodnotu
c) pre daný kľúč vypočíta adresu
d) vracia pre dva rovnaké kľúče rôznu hodnotu

3. Vektor, v ktorom je uložená rozptylová (hashovacia) tabuľka, vyzerá pri použití rozptylovej funkcie h(k) = k mod 5, metódy lineárneho skúšania (linear probing) a po vložení kľúčov 6, 5, 9, 4 (vkladaných v poradí zľava doprava) takto:

a)

0	1	2	3	4
5	6	4		9

b)

0	1	2	3	4
5	6	9		4

c)

0	1	2	3	4
4	6	5		9

d)

0	1	2	3	4
4	5	6		9

4. Uvažujte tento fragment algoritmu v pseudojazyku:

```
declare zásobník znakov
while ( je znak na vstupe )
{
   prečítaj znak
   vlož/push znak do zásobníka
}
while ( zásobník nie je prázdny )
{
   vyber/top,pop znak zo zásobníka
   zobraz znak
}
```

Čo sa zobrazí po vykonaní tohto fragmentu, ak je na vstupe postupnosť znakov deneb?

a) dnb
b) bened
c) deneb
d) bbeenneedd

5. Obsah ktorého poľa reprezentuje binárnu haldu? (práve jeden) Haldu nakreslite v tvare binárneho stromu.

☐ A =

0	1	2	3	4	5	6	7	8	9	10	11
3	5	15	7	18	22	35	30	9	17		

☐ B =

0	1	2	3	4	5	6	7	8	9	10	11
3	5	18	15	7	22	35	30	9	17		

☐ C =

0	1	2	3	4	5	6	7	8	9	10	11
3	5	18	7	15	22	35	30	9	17		

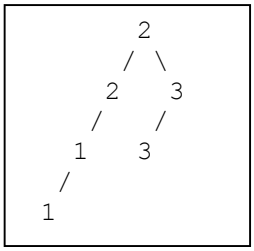
Z haldy teraz odstáňte najmenší prvok vrátane obnovenia haldovitosti. Napíšte výslednú haldu:

0	1	2	3	4	5	6	7	8	9	10	11

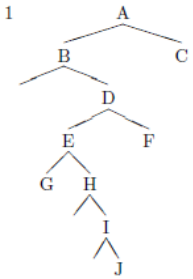
6. Naprogramujte transformáciu binárneho vyhľadávacieho stromu do „zdvojeného“ binárneho vyhľadávajúceho stromu (BVS). Ku každému vrcholu v BVS treba vytvoriť jeho kópiu a pridať ju ako ľavý potomok pôvodného vrchola. Výsledný strom má stále byť BVS (s tým, že sa pripúšťajú vrcholy s rovnakým ohodnotením a budú vľavo).

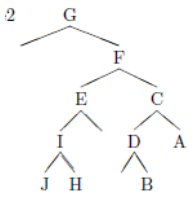
Napríklad BVS:

sa transformuje na



7. Uvažujte dva stromy:





A. Preorder, postorder

B. Postorder, inorder

C. Postorder, postorder

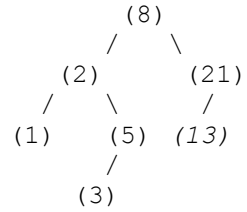
D. Inorder, inorder

E. Postorder, preorder

Aké prehľadávanie prvého stromu a aké prehľadávanie druhého stromu vytvoria rovnaký výstup (tj postupnosť znakov vrcholov)? Príslušnú postupnosť aj napíšte.

8. Dokážte (sporom), že ak má vrchol v binárnom vyhľadávacom strome dva priame potomky, tak jeho (inorder) nasledovník nemá ľavý potomok.

Pomôcka. Všimnime si, že v tomto BVS má vrchol s kľúčom 2 dva potomky. Jeho nasledovník je vrchol s kľúčom 3. Vrchol s kľúčom 3 nemá ľavý potomok. Mohol by mať nanajvýš pravý potomok.



Návod. Predpokladajte, že dôsledok vety neplatí. Označme x vrchol, ktorý má dva potomky. Označme jeho nasledovník s. Predpokladajme teda, že s má ľavý potomok. Pokračujte logicky na seba nadväzujúcimi krokmi, ktoré povedú ku sporu. Riešenie pozostáva z 3 krokov.

9. Uvažujte nižšie napísanú implementáciu algoritmu usporadúvania zlučovaním (MergeSort):

```
public void mergeSort(int[] array, int l, int r) {
     if (l < r) {
          int m = (l + r) / 2;
          mergeSort( array , l , m );
          mergeSort( array , m + 1 , r );
          merge( array , l , m , r );
     }
}

public void merge(int[] array, int l, int m, int r) {
     int[] temp = array.clone();
     int left = l;
     int right = m + 1;
     int pos = left;

     while (left <= m || right <= r) {
          if (left <= m && right <= r) {
               if (temp[ left ] < temp[ right ]) {
                    array[ pos ] = temp[ left ];
                    left++;
               } else {
                    array[ pos ] = temp[ right ];
                    right++;
               }
          } else if (left <= m) {
               array[ pos ] = temp[ left ];
               left++;
          } else if (right <= r) {
               array[ pos ] = temp[ right ];
               right++;
          }
          pos++;
     }
}
```

Je takto implementovaný algoritmus stabilný?

a) Áno, lebo:
b) Nie, lebo:

Vo odôvodnení odpovede uveďte, ktorý príkaz/y zaručia/môžu porušiť stabilitu. V prípade odpovede Nie uveďte aj príklad postupnosti, pri usporadúvaní ktorej dôjde k nestabilite.

10. Uvažujte algoritmus usporadúvania spočítavaním (Counting sort). Uvažujte, že údaje sú celé jednociferné čísla. Používa tri polia: A – v ňom je pôvodná postupnosť n prvkov, B - v ňom bude usporadaná postupnosť a C – tzv. počítadlá.

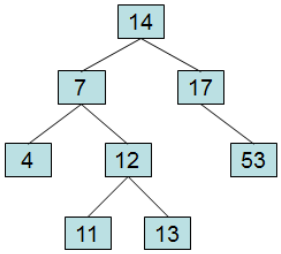
a) Doplňte dĺžky jednotlivých polí: A[n], B[ ], C[ ].
b) Uvažujte vstupnú postupnosť 1, 4, 1, 2, 9, 5, 2. Napíšte, aký bude obsah všetkých troch polí na konci.

A: B: C:

11. Je dané usporiadané pole rôznych celých čísel A[1..n]. Navrhnite algoritmus, ktorý rozhodne, či existuje index i taký, že i=A[i]. Ak existuje, najdiindex(A,1,n) vráti i, ak nie, vráti 0. Vaše riešenie musí byť rýchlejšie ako O(n) a nesmie vyžadovať viac dodatočnej pamäti ako O(1).

12. Uvažujte AVL strom:

Pripíšte ku každému uzlu hodnotu jeho faktoru vyváženia. Vložte do tohto stromu 8 a ak treba, urobte úpravy, aby bol výsledný strom opäť AVL. Nakreslite aj strom po každej zmene.



13. Ukážte priebeh radixového usporadúvania (radix je 10) tejto postupnosti čísel:

```
36 9 0 25 1 49 64 16 81 4
```

14. Daná je postupnosť n čísel a jedno číslo, ktoré nazveme s. Navrhnite algoritmus, ktorý rozhodne v lineárnom čase O(n), či súčet niektorých dvoch čísel v postupnosti je s. Nemusíte písať presne v nejakom programovacom jazyku, stačí pseudokód alebo slovný opis.

15. Uvažujte haldu:

a) Nakreslite ju po pridaní 3.
b) Nakreslite ju po odobratí minima (z pôvodnej, ako bola pred vložením 3).