Programovanie (1) v C/C++, Test pre pokročilých, 23. 9. 2020

Meno a priezvisko: .....

/8	/6	/4	/6	/8	/8	$\sum$ (/40)

POKYNY: Odpovede píšte priamo na miesto vyhradené v zadaní. Ak potrebujete viac priestoru, vypýtajte si ďalšie listy papiera. Nezabudnite každý ďalší list podpísať.

1 Napíšte program, ktorý zo štandardného vstupu prečíta celé číslo $n \geq 1$ a n reálnych čísel $a_0, \ldots, a_{n-1}$. Na výstup potom vypíše *modus* reálnych čísel $a_0, \ldots, a_{n-1}$ – čiže hodnotu, ktorá sa medzi týmito číslami vyskytuje najčastejšie. V prípade, že existuje viac ako jeden modus, vypíšte najmenší z nich.

Príklady vstupu a výstupu:

```
Vstup:                  Výstup:
7                       2
2 -1 3.14 -1 2 0 2      2
```

```
Vstup:                  Výstup:
6                       1.5
2 1.5 2 1.5 2 1.5       1.5
```

2 Uvažujme nasledujúcu funkciu `L`, v ktorej tele sa volá rekurzívna funkcia `produce` vypisujúca na štandardný výstup nejaký text.

```
void produce(char c, int n) {
    if (n <= 0) {
        cout << c;
    } else if (c == 'a') {
        produce('a', n - 1);
        produce('b', n - 1);
    } else {
        produce('a', n - 1);
    }
}

void L(char *ax, int n) {
    for (int i = 0; ax[i] != 0; i++) {
        produce(ax[i], n);
    }
}
```

Zistite, aký text sa vypíše po volaní funkcie `L` na nasledujúcich troch vstupoch (vždy je najprv v úvodzovkách uvedený reťazec `ax` a za ním nasleduje číslo `n`):

Vstup: `"a" 2`
Odpoveď:

Vstup: `"a" 4`
Odpoveď:

Vstup: `"bba" 3`
Odpoveď:

3 Preveďte infixový výraz `3 + (2 * 7 - 9) * 2 + 2 / (1 - 3)` do prefixovej notácie.

4 Nižšie je kostra funkcie `llcopy`, ktorá skopíruje spájaný zoznam `list1` do spájaného zoznamu `list2`. Kopírovanie prebieha na úrovni hodnôt – po vykonaní funkcie sú v zozname `list2` v rovnakom poradí uložené rovnaké hodnoty ako v zozname `list1`, ale žiaden z uzlov zoznamu `list2` nie je rovný uzlu zo zoznamu `list1`. Doplňte do kostry chýbajúce časti tak, aby funkcia túto úlohu realizovala správne. Kód dopĺňajte iba na vyznačené miesta.

```
struct node {
    int data;
    node *next;
};

struct linkedList {
    node *first;
};

void llcopy(_____ list1, _____ list2) {
    if (list1.first == NULL) {

        list2.first = _____;
    } else {
        list2.first = _____;

        _____ p1 = list1.first;

        _____ p2 = list2.first;

        while (_____) {
            p2->data = p1->data;

            if (_____ == NULL) {
                p2->next = NULL;
            } else {

                p2->next = _____;
            }
            p1 = _____;

            p2 = _____;
        }
    }
}
```

5 Uvažujme binárne vyhľadávacie stromy realizované štruktúrou `binarySearchTree` nasledovne:

```
struct node {
    int key;

    node *parent;
    node *left;
    node *right;
};

struct binarySearchTree {
    node *root;
};
```

Predpokladajme, že môže nastať situácia, keď si nie sme istí, či do binárneho vyhľadávacieho stromu niekto manuálne nezasahoval. Keďže po takomto zásahu už strom nemusí naďalej spĺňať definíciu binárneho vyhľadávacieho stromu, zíde sa funkcia, ktorá zistí, či je strom reprezentovaný danou premennou typu `binarySearchTree` korektný, t.j. či spĺňa definíciu binárneho vyhľadávacieho stromu. Napíšte funkciu s hlavičkou

```
bool bstValidate(binarySearchTree &t);
```

ktorá realizuje túto úlohu – pre vstupný strom `t` teda vráti `true` práve vtedy, keď ide o korektný binárny vyhľadávací strom. (Strom `t` považujeme za korektný aj v prípade, že `t.root` je `NULL`.)

6 Uvažujme dátovú štruktúru radu (t.j. frontu) celých čísel, ktorá podporuje nasledujúce operácie:

```
/* Inicializuje prazdny rad */
void init(queue &q);

/* Zisti, ci je rad prazdny */
bool isEmpty(queue &q);

/* Prida prvok item na koniec radu */
void enqueue(queue &q, int item);

/* Odoberie prvok zo zaciatku radu a vrati jeho hodnotu */
int dequeue(queue &q);

/* Vrati prvok zo zaciatku radu, ale necha ho v rade */
int peek(queue &q);

/* Uvolni pamat */
void destroy(queue &q);
```

Predpokladajme, že rad `q` už obsahuje niekoľko *kladných* celých čísel. Napíšte kus programu, ktorý *bez pomoci ďalších dátových štruktúr* z radu `q` vymaže všetky nepárne čísla a zvyšné prvky radu `q` následne uvedie do rovnakého poradia ako na začiatku. Napríklad z radu `2 1 5 4 9 8 4` by sa po vykonaní vášho programu mal stať rad `2 4 8 4`. Váš program by mal okrem radu `q` používať už iba premenné primitívnych typov. *Pomôcka:* keďže predpokladáme kladnosť všetkých pôvodných prvkov radu, možno nulu alebo záporné čísla využiť na pomocné účely.