პრაქტიკული 8: BST: სიმაღლე, ძებნა

სააუდიტორიო სამუშაო:

- ამოცანა 1 /ხის შემოვლა/
- ამოცანა 2 /ძებნა/ >>>
- სავარჯიშოები >>>

ამოცანა 1. შემდეგი მონაცემებისგან ავაგოთ ძებნის ორობითი ხე: 47 51 23 15 40 49 21. ჩავთვალოთ, რომ პირველი მონაცემი მოთავსებულია a_1 მისამართზე, მეორე a_2-ზე, და ა.შ. შემდეგ, მოვიყვანოთ ხის შემოვლის ალგორითმის ფსევდოკოდი და ვაჩვენოთ მისი მუშაობა (ტრასირება) მოცემულ ხეზე შესრულებული სტრიქონების მიმდევრობის მითითებით.

ამოხსნა: ხის შემოვლის ალგორითმი, რომელიც ზრდადობით ბეჭდავს გასაღებებს, არის:

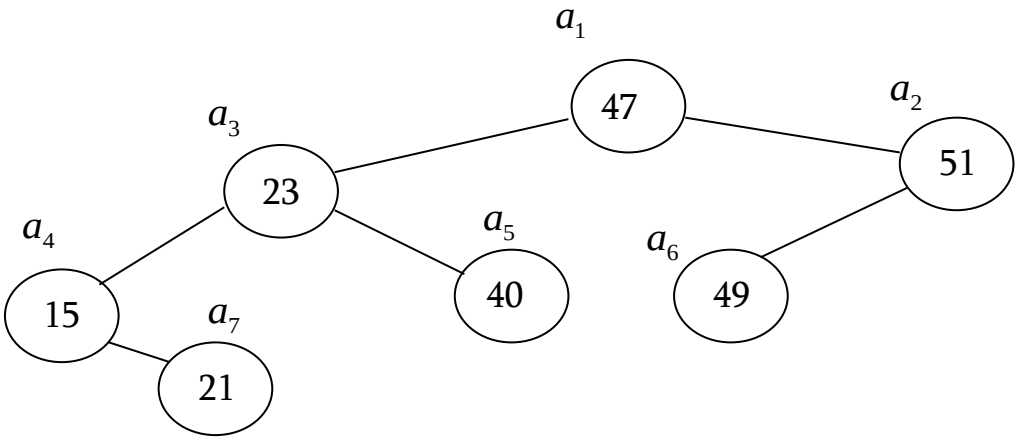
```
void inorder_walk(link x)
      if (x != nullptr)
            ST_inorder_walk(left(x));
            cout << key(x) << endl;
            ST_inorder_walk(right(x));
```

ვნახოთ თუ როგორ მუშაობს იგი ბიჯების მიხედვით:

```
inorder_walk(x = a1)
    inorder_walk(x = a3)
        inorder_walk(x = a4)
            inorder_walk(x = nullptr)
            cout << 15 << endl;
            inorder_walk(x = nullptr)
        cout << 23 << endl;
        inorder_walk(x = a5)
            inorder_walk(x = nullptr)
            cout << 40 << endl;
            inorder_walk(x = nullptr)
    cout << 47 << endl;
    inorder_walk(x = a2)
        inorder_walk(x = a6)
            inorder_walk(x = nullptr)
            cout << 49 << endl;
            inorder_walk(x = nullptr)
        cout << 51 << endl;
        inorder_walk(x = nullptr)
```

<<< **ამოცანა 2.** შემდეგი მონაცემებისგან ავაგოთ ძებნის ორობითი ხე: 47 51 23 15 40 49 21. ჩავთვალოთ, რომ პირველი მონაცემი მოთავსებულია a_1 მისამართზე, მეორე a_2-ზე, და ა.შ. შემდეგ, ვაჩვენოთ `ST_search(`a_1`, 21)` ალგორითმის მუშაობა ბიჯების მიხედვით. ვისარგებლოთ ფსევდოკოდით, რომელშიც ტიპი `link` აღნიშნავს წვეროს მისამართს:

```
link ST_search (link x, int k)
{
   if (nullptr == x || k == key(x)) return x;
   if (k < key(x)) return ST_search (left(x) , k) ;
   return  ST_search (right(x) , k) ;
}
```




```
ST_search(x = a1, k = 21)= a7
  if (nullptr == a1 || 21 == 47)☹
  if (21 < 47 ) ☺
      return ST_search (x = a3, k = 21)= a7          //left(a1)=a3
            if (nullptr == a3 || 21 == 23)☹
            if (21 < 23 )  ☺
                return ST_search (x = a4, k = 21)= a7     //left(a3)=a4
                      if(nullptr ==a4 || 21 == 15)☹
                      if (21 < 15 )  ☹
                      return ST_search (x = a7, k = 21)= a7
                            if (nullptr ==a1 || 21= 21)return a7
```

<<< სავარჯიშოები:

1. შემდეგი მონაცემებისგან:

   ა) 7 51 23 115 4 1 ბ) 55 23 51 53 20

   ააგეთ ძებნის ორობითი ხე. ჩათვალეთ, რომ პირველი მონაცემი მოთავსებულია a_1 მისამართზე, მეორე a_2-ზე, და ა.შ.

   მოიყვანეთ ხის შემოვლის ერთ-ერთი ალგორითმის ფსევდოკოდი და აჩვენეთ მისი მუშაობა (ტრასირება) მოცემულ ხეზე შესრულებული სტრიქონების მიმდევრობის მითითებით.

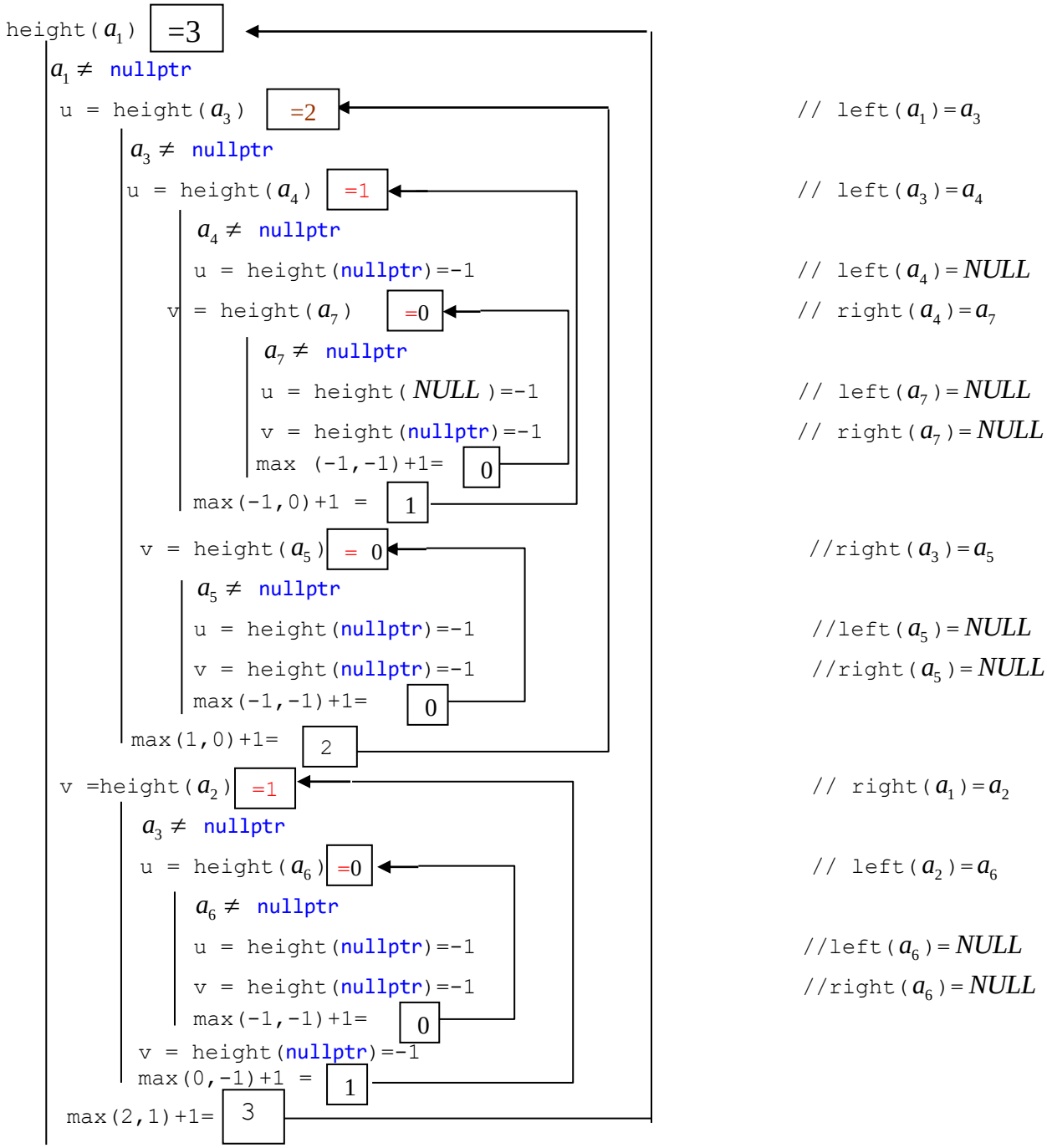
   მოიყვანეთ ხის სიმაღლის გამოთვლის ალგორითმის ფსევდოკოდი და აჩვენეთ მისი მუშაობა (ტრასირება) მოცემულ ხეზე შესრულებული სტრიქონების მიმდევრობის მითითებით.

   შემდეგ, აჩვენეთ `ST_search(`a_1`,115)` ალგორითმის მუშაობა (ტრასირება) ფსევდოკოდის სტრიქონების მიმდევრობის მითითებით.

2. შემდეგი მონაცემებისგან ავაგოთ ძებნის ორობითი ხე: 47 51 23 15 40 49 21. ჩავთვალოთ, რომ პირველი მონაცემი მოთავსებულია a_1 მისამართზე, მეორე a_2-ზე, და ა.შ. ხის სიმაღლის გამოთვლის ალგორითმის ფსევდოკოდია

```
int  height (link x)
  if (x == nullptr) return -1;
  int u = height (left(x));
  int v = height (right(x));
  return ( u < v )?( v +1):(u +1);
```

ხის სიმაღლის გამოთვლის ალგორითმის მუშაობას ბიჯების მიხედვით აქვს სახე::



=0