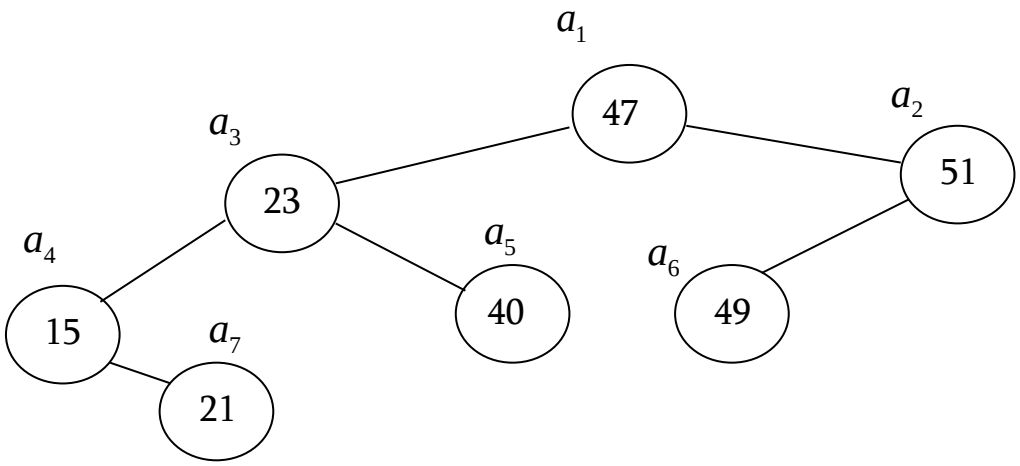
პრაქტიკული 9: BST: ჩამატება, წინა და მომდევნო კვანძები

სააუდიტორიო სამუშაო:

- ამოცანა 1 /კვანძის დამატება/
- ამოცანა 2 /მომდევნო კვანძის განსაზღვრა/ >>>
- სავარჯიშოები >>>

ამოცანა 1. დავხატოთ შემდეგი მონაცემებისგან აგებული 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, ჩავამატოთ ახალი `z` კვანძი გასაღებით 25. ვაჩვენოთ `ST_insert(`a_1`,z)` ალგორითმის მუშაობა ბიჯების მიხედვით.

ამოხსნა: ვისარგებლოთ ფსევდოკოდით, რომელშიც ტიპი `link` აღნიშნავს წვეროს მისამართს. შესაბამის ხეს აქვს სახე:




```
link ST insert(link root, link z){
      1       link x,y;
      2       y = nullptr;
      3       x = root;
      4       while (nullptr != x ) {
      5           y = x;
      6           if   ( key(z) < key(x))   x = left(x);
      7           else  x= right(x);
      8       }
      9       p(z) = y;
      10      if (nullptr == y) root = z;
      11      else  {
      12          if   (key(z) <  key(y))  left(y) = z;
      13          else  right(y) = z;
      14      }
      15      return root;
          }
```

ვაჩვენოთ ალგორითმის მუშაობა სტრიქონების მიმდევრობის მითითებით:

`ST_insert(`a_1`,z)`

```
y = nullptr;
x = a1;
while ( a1 ≠ nullptr)
    y = a1;
    if(25<47)    x=a3;
```

```
while ( a3 ≠ nullptr)
    y = a3;
    if(25<23)   ☹
    else x=a5;
while ( a5 ≠ nullptr)
    y = a5;
    if(25<40)   x=NULL;
while (nullptr!= nullptr) ☹
p(z) = a5;
if (nullptr == y) ☹
else
    if   (25 <  40)  left(y) = z;    // a5 მისამართზე left გახდა z
return root;
```

<<< ამოცანა 2. წინა ამოცანის ხისთვის, რისი ტოლია a_1-ის და a_5 -ის მომდევნო (გასაღებების სიდიდის აზრით) კვანძების მისამართები?

ამოხსნა: ვისარგებლოთ ფსევდოკოდით,

```
link ST_successor (link x){
    if (nullptr!= right(x) ) return ST_minimum ( right(x) );
    while (nullptr!= p(x) && x == right(p(x))) {
        x = p(x);
    }
    return p(x);
}
```

ვაჩვენოთ ალგორითმის მუშაობა სტრიქონების მიმდევრობის მითითებით:

```
ST_ successor(x = a1)
    if (nullptr!= a2 ) return ST_minimum(a2)=a6;

ST_ successor(x = a5)
    if (nullptr!= nullptr) ☹
    while (nullptr!= p(a5)=a3 && a5 == right(a3)) {
        x = a3;    // x და არა a5, რადგან a5 არის ე.წ. right-value
    while (nullptr!= p(a3)=a1  && a3 == right(a1))☹
    return  a1(= p(a3));
```

<<< სავარჯიშოები: შემდეგი მონაცემებისგან:

1.	7	51	23	115	4	1
2.	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 ახალი z კვანძის ჩამატების ალგორითმის ფსევდოკოდი და აჩვენეთ მისი მუშაობა (ტრასირება) მოცემულ ხეზე შესრულებული სტრიქონების მიმდევრობის მითითებით, თუ key(z) = 22 და key(z) = 48. შემდეგ, ახლად ჩასმული კვანძების მაგალითზე აჩვენეთ წინა და მომდევნო კვანძების განსაზღვრის ალგირითმის მუშაობა შესრულებული სტრიქონების მიმდევრობის მითითებით.