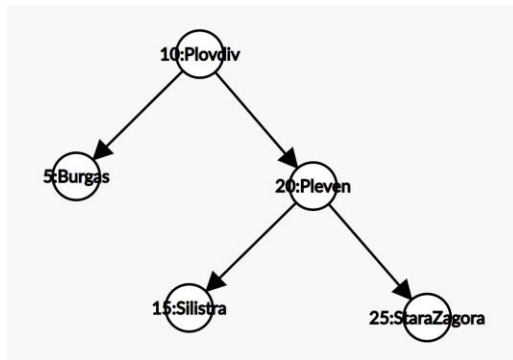


Задачи:

За следващите задачи се иска да създадете клас **BinaryTree**, който ще представлява двоично наредено дърво ☺

1. Създайте конструктор на **BinaryTree**, който приема като параметър сортиран масив от двойки ключ-стойност.
2. Реализирайте метод **prittyPrint()**, който да извежда дървото на екрана по подходящ начин.
3. Реализирайте предикат **contains(T key)**, който проверява дали ключа key се съдържа в дървото.
4. Реализирайте метод **calculateHeight()**, който пресмята височината на дървото.
5. Реализирайте метод **sumLeaves()**, който връща сумата от ключовете на листата на дървото.
6. Реализирайте метод **countLeaves()**, който връща броя на всичките му листа.
7. Реализирайте метод **remove(T x)**, който премахва първото срещане на възела с ключ x.
8. Реализирайте метод **reduce(T (*operation)(T, T), T accumulator)**, който прилага operation върху ключовете на дървото в ред **ЛКД**.
9. Реализирайте метод **serializeTree(std::ofstream& out)**, за сериализация на дървото по следният начин (**Scheme format**):
 - Празното дърво се представя като **"()"**
 - Нека е дадено дървото t с корен x:y, ляво поддърво tL и дясно поддърво tR. Ако sL е представянето в "Scheme format" на tL, а sR – на tR, то низът **"(x:y sL sR)"** е представянето на дървото t, където "x", "sL" и "sR" са съответните низове.

Пример:



се представя като

(10:Plovdiv (20:Pleven (25:StaraZagora () ()) (15:Silistra () ())) (5:Burgas () ()))

10. Даден е вектор **v** от цели числа. Казваме, че двоичното дърво с положителни числа по върховете **t** представя **v** при следните условия:
 - **v** е празният вектор и **t** е празното дърво
 - ако **v = v[0],...,v[k-1]**, а **m = [k / 2]** (долна цяла част), то коренът на **t** съдържа числото **v[m]**, лявото поддърво на **t** представя вектора **v[0],...,v[m-1]**, а дясното поддърво на **t** представя вектора **v[m+1],...,v[k-1]**.

Забележка: ако **k = 2**, то десния подвектор считаме за празен.

- а. Да се реализира функция, която построява дърво, представящо вектора **v**, и връща указател към корена му.

Упътване: Ако v е вектор, то с помощта на следния конструктор `std::vector<unsigned>` `L(v.begin(), v.begin() + count)` ще получите първите `count` елемента от v , а с `std::vector<unsigned> R(v.begin() + start, v.end())`, ще получите суфикса на v , започващ от елемента с индекс `start`.

Пример:

```
std::vector<unsigned> v{1, 2, 3, 4, 5, 6},  
    L(v.begin(), v.begin() + 3),  
    R(v.begin() + 4, v.end());
```

Тогава L е векторът 1, 2, 3, а R е векторът 5, 6.

- b. Да се реализира функция, която връща вектор, съставен от възлите по път в дървото от корен до листо, в който сумата на елементите е максимална.