```python
from queuelinked import LinkedQueue

class BinarySearchTree:
    class _Node:
        __slots__ = '_element','_left','_right'

        def __init__(self,element, left=None, right=None):
            self._element = element
            self._left = left
            self._right = right

    def __init__(self):
        self._root = None
        self._size = 0

    def insert(self, e):
        troot = self._root
        ttroot = None
        while troot:
            ttroot = troot
            if e < troot._element:
                troot = troot._left
            elif e > troot._element:
                troot = troot._right
        node = self._Node(e)
        if self._root:
            if e < ttroot._element:
                ttroot._left = node
            else:
                ttroot._right = node
        else:
            self._root = node

    def recurinsert(self, troot, e):
        if troot == None:
            node = self._Node(e)
            return node

        if e < troot._element:
            troot._left = self.recurinsert(troot._left, e)
        elif e > troot._element:
            troot._right = self.recurinsert(troot._right, e)

        return troot



    def search(self, k):
        troot = self._root
        while troot:
            if k < troot._element:
                troot = troot._left
            elif k > troot._element:
                troot = troot._right
            else:
                return True
        return False

    def levelorder(self):
        Q = LinkedQueue()
        t = self._root
        print(t._element,end='--')
        Q.enqueue(t)

        while not Q.is_empty():
            t = Q.dequeue()
            if t._left:
                print(t._left._element, end='--')
                Q.enqueue(t._left)
            if t._right:
                print(t._right._element, end='--')
                Q.enqueue(t._right)

    def inorder(self, troot):
        if troot:
            self.inorder(troot._left)
            print(troot._element, end='--')
            self.inorder(troot._right)

    def preorder(self,troot):
        if troot:
            print(troot._element,end='--')
            self.preorder(troot._left)
            self.preorder(troot._right)

    def postorder(self, troot):
        if troot:
            self.postorder(troot._left)
            self.postorder(troot._right)
            print(troot._element, end='--')


B = BinarySearchTree()
```

```
B._root = B.recurinsert(None,70)
B.recurinsert(B._root,30)
B.recurinsert(B._root,90)
B.recurinsert(B._root,40)
B.recurinsert(B._root,50)
B.recurinsert(B._root,110)
B.inorder(B._root)
print()
print(B.search(25))

```