DATE: PAGE:

B-trees:

```
class BTreeNode:
    def __init__(self, leaf=False):
        self.leaf = leaf
        self.keys = []
        self.child = []

class Btree:
    def __init__(self, t):
        self.root = BtreeNode(True)
        self.t = t

    def search_key(self, k, x=None):
        if x is not None:
            i = 0
            while i < len(x.keys) and k > x.keys[i][0]:
                i += 1
            if i < len(x.keys) and k == x.keys[i][0]:
                return (x, i)
            elif x.leaf:
                return None
            else:
                return self.search_key(k, x.child[i])
        else:
            return self.search_key(k, self.root)
```

DATE: PAGE:

```
def insert_key(self, k):
    root = self.root
    if len(root.keys) == (2*self.t) - 1:
        temp = BTreeNode()
        self.root = temp
        temp.child.insert(0, root)
        self.split(temp, 0)
        self.insert_non_full(temp, k)
    else:
        self.insert_non_full(root, k)

def insert_non_full(self, x, k):
    i = len(x.keys) - 1
    if x.leaf:
        x.keys.append((None, None))
        while i >= 0 and k[0] < x.keys[i][0]:
            x.keys[i+1] = x.keys[i]
            i -= 1
        x.keys[i+1] = k
    else:
        while i >= 0 and k[0] < x.keys[i][0]:
            i -= 1
        i += 1
        if len(x.child[i].keys) == (2*self.t) - 1:
            self.split(x, i)
            if k[0] > x.keys[i][0]:
                i += 1
        self.insert_non_full(x.child[i], k)
```