```cpp
#include <iostream>
using namespace std;

//
//顺序查找
//顺序表
int Search(int arr[], int n, int key)
{
    int i;
    for (i = 0; i < n; i++)
        if (arr[i] == key)
            return 1;
    return -1;
}
//链式表
LNode* Search(LNode* head, int key)
{
    LNode* p = head->next;
    while (p != NULL)
    {
        if (p->next == key)
            return p;
        p = p->next;
    }
    return NULL;
}
//折半查找(适用有序数组)
int BSearch(int arr[], int low, int high, int key)
{
    while (low <= high)
    {
        int mid = (low + high) / 2;
        if (arr[mid] == key)
            return mid;
        else if (arr[mid] > key)
            high = mid - 1;
        else
            low = mid + 1;
    }
    return -1;
}
//分块查找(适用含有一定有序性的数组,分块，利用折半查找确定范围，再利用顺序查找找到元
  素)
typedef struct
{
    int maeKey;
    int low, high;

}indexElem;
indexElem index[5];
int keys[15];
//二叉排序树
//查序1.
BTNode* BSTSearch(BTNode* p, int key)
{
    while (p != NULL)
    {
        if (key == p->key)
            return p;
        else if (key < p->key)
            p = p->lChild;
        else
            p = p->rChild;
    }
    return NULL;
}
//2.
BTNode* BSTSearch(BTNode* p, int key)
{
    if (p == NULL)
        return NULL;
    else
    {
        if (p->key == key)
            return p;
        else if (key < p->key)
            return BSTSearch(p->lChild, key);
        else
            return BSTSearch(p->rChild, key);
    }
}
//插入关键字
int BSTInsert(BTNode*& p, int key)
{
    if (p == NULL)
    {
        p = (BTNode*)malloc(sizeof(BTNode));
        p->lChild = p->rChild = NULL;
        p->key = key;
        return 1;

    }
    else
    {
        if (key == p->key)
            return 0;
        else if (key < p->key)
            return BSTInsert(p->lChild, key);
        else
            return BSTInsert(p->rChild, key);
    }
}
//删除关键字
//具体情况具体分析吧

int main()
{

}
```