习题1

改写后的源代码文件名为1.c0

证明代码中的三个循环不变量：

//@loop\_invariant lower >= left && lower<=upper && upper<=right;

//@loop\_invariant (left==0&&lower==left)||A[lower-1]<x;

//@loop\_invariant (right==n-1&&upper==right)||A[upper+1]>x;

1. 在初始情况下，有lower = left和upper = right，而且函数前置条件：

left >= 0 && left <= right && right < \length(A)

所以，循环不变量(1)、(2)和(3)都成立。

2. 假设在某次进入循环前，3个循环不变量都成立，下证明经过一次循环处理后，三个循环不变量是否仍然成立。

mid = lower + (upper-lower)/2 。

现在分三种情况进行讨论：

(1) A[mid] = x：

这种情况是找到了x，函数将返回到被调用的地方，不需要判断循环不变量是否成立；尽管如此，由于在这种情况下lower和upper都没有改变，所以循环不变量(1)、(2)和(3)仍然成立。

(2) A[mid] < x：那么，new lower = mid + 1，new upper = upper

由代码中mid计算公式、我们的假设和初级算术规则，得到第一个循环不变量left<=lower<=upper<=right成立。

对于第二个循环不变量，有A[new lower-1]=A[mid]<x,成立。

对于第三个循环不变量，因为new upper =upper，所以保持成立。

(3) A[mid] > x：

证明同（2）

习题2

源代码文件名为2.c0

当//@loop\_invariant is\_in(x, A, 0, n) == is\_in(x, A, lower, upper);

若is\_in(x, A, 0, n)为否，由于//@loop\_invariant lower >=0 && lower<=upper && upper<=n;显然is\_in(x, A, lower, upper)也为否；

下面只用考虑为真的情况

初次进入循环时：若is\_in(x, A, 0, n)为真，显然is\_in(x, A, lower, upper)也为真；

是亦然。

假设在某次进入循环前，循环不变量都成立

(1) A[mid] = x：

这种情况是找到了x，函数将返回到被调用的地方，不需要判断循环不变量是否成立；尽管如此，由于在这种情况下lower和upper都没有改变，所以循环不变量(1)、(2)和(3)仍然成立。

(2) A[mid] < x：new lower = mid + 1，new upper = upper。由于//@requires is\_sorted(A, 0, n);则upper到mid的所有数组值都<x，则is\_in(x, A, new lower, new upper)必然为真

(3) A[mid] > x：

证明同（2）

习题3

举例：所给数组A共有100个元素，值全都一样，此时将返回50。

源代码文件名为3.c0.循环不变量的证明与习题1类似，再找到相等的点后，将一直向左寻找最初相等的点并返回

改写后对于上述输入将返回0。

习题4

//@assert lower<=mid && mid<=upper;

上述断言标签会注意到有瑕疵。因为此时lower+upper越界后会导致mid<lower，断言将终止程序执行，当upper + lower >= 2^31时。

练习5

    while (lower < upper)

    //@loop\_invariant 0 <= lower && lower <= upper && upper <= n;

    //@loop\_invariant (lower == 0 || A[lower-1] < x);

    //@loop\_invariant (upper == n || A[upper] > x);

    {

        lower = lower;

        upper = upper;

        int mid = lower + (upper - lower) / 2;

        //@assert lower <= mid && mid < upper;

        if (A[mid] == x)

            return mid;

        else if (A[mid] < x)

            lower = mid + 1;

        else /\*@assert(A[mid] > x);@\*/

            upper = mid;

    }

是把代码换成这样子？没看懂题意，没发现什么问题。