1、顺序查找和折半查找，对初始线性表有什么样的要求。

1）答:顺序查找是针对以顺序表或者链式表表示的静态查找表;

2）折半查找是针对有序表表示的静态查找表,且限于顺序存储结构。

2、折半查找的效率如何？

答：一般情况下，表长为 n 的折半查找的判定树的深度和含有 n 个结点的完全二叉树的深度相同。假设 $n=2^h-1$ 并且查找概率相等

则 $$ASL_{bs}=\frac{1}{n}\sum_{i=1}^{n}C_i=\frac{1}{n}\left[\sum_{j=1}^{h}j\times 2^{j-1}\right]=\frac{n+1}{n}\log_2(n+1)-1$$

在 n>50 时，可得近似结果 $ASL_{bs}\approx\log_2(n+1)-1$

3、二叉排序树或者是一棵空树；或者是具有什么样特性的二叉树？

答：（1）若它的左子树不空，则左子树上所有结点的值均小于根结点的值

（2）若它的右子树不空，则右子树上所有结点的值均大于根结点的值；

（3）它的左、右子树也都分别是二叉排序树。

4、**二叉平衡树**是二叉查找树的另一种形式，其特点是什么？

答：**树中每个结点的左、右子树深度之差的绝对值不大于 1**。

```
#define KeyType int
#define InfoType char

#define SSTABLE_INIT_SIZE 100

typedef struct{
    KeyType key;
    InfoType info;
}ElemType;

typedef struct{
    ElemType *elem;
    int      length;
}SSTable;

int Search_Seq(SSTable ST,KeyType kval)
{
    int i;

    ST.elem[0].key=kval;
```

```
    for(i=ST.length;i>=0;i--)
    {
        if(ST.elem[i].key==    kval    )
        break;
    }
    return    i    ;
}/*Search_Seq*/

main()
{
    SSTable ST;
    KeyType kval;
    int k;

    ST.elem=(ElemType*)malloc(sizeof(ElemType)*SSTABLE_INIT_SIZE);

    ST.elem[0].key=0;
    ST.elem[1].key=64;
    ST.elem[2].key=80;
    ST.elem[3].key=13;
    ST.elem[4].key=56;
    ST.elem[5].key=37;
    ST.elem[6].key=92;
    ST.elem[7].key=19;
    ST.elem[8].key=05;
    ST.elem[9].key=88;
    ST.elem[10].key=21;
    ST.elem[11].key=75;

    ST.length=11;

    printf("\n\n");

    printf("input the value to be searched:");
    scanf("%d",&kval);

    k=Search_Seq(  ST  ,kval);

    printf("\n\n");

    if(k==0)
      printf("sorry,not searched!");
    else
```

```
        printf("searched and the position is:%d",k);
}
```

附 2：原程序

```
#define KeyType int
#define InfoType char

#define SSTABLE_INIT_SIZE 100

typedef struct{
      KeyType key;
      InfoType info;
}ElemType;

typedef struct{
      ElemType *elem;
      int          length;
}SSTable;

int Search_Bin(SSTable ST,KeyType kval)
{
      int low,high,mid;

    low=1;
    high=ST.length;

    while(low<=high){
          mid=(      low+high      )/2;
          if(kval==ST.elem[mid].key) return mid;
          else
                if(kval<ST.elem[mid].key)
                      high=mid-1;
                else
                      low=      mid+1      ;
    }/*while*/
    return 0;

}/*Search_bin*/

main()
{
      SSTable ST;
```

```
    KeyType kval;
    int k;

    ST.elem=(ElemType*)malloc(sizeof(ElemType)*SSTABLE_INIT_SIZE);

    ST.elem[0].key=0;
    ST.elem[1].key=05;
    ST.elem[2].key=13;
    ST.elem[3].key=19;
    ST.elem[4].key=21;
    ST.elem[5].key=37;
    ST.elem[6].key=56;
    ST.elem[7].key=64;
    ST.elem[8].key=75;
    ST.elem[9].key=80;
    ST.elem[10].key=88;
    ST.elem[11].key=92;

    ST.length=11;

    printf("\n\n");

    printf("input the value to be searched:");
    scanf("%d",&kval);

    k=Search_Bin(ST,kval);

    printf("\n\n");

    if(k==0)
   printf("sorry,not searched!");
    else
   printf("searched and the position is:%d",k);
}
```