本程序为在 L 型公理下对正确命题给出自动证明的程序，对于正确的命题，一定在给出证明后停机。

本程序采用自顶向下的搜索算法，求出一个命题的“最短”的直接证明，前提是该命题是可证的（在命题逻辑中，这等价为该命题为真）。

注意：

这里“最短”的附加条件是，证明中所有的定理仅使用一次，哪怕是完全一样的命题，使用多次也需要多次重复证明。这大大简化了搜索，一个命题的“最短”证明中包含的的子定理的证明一定是“最短”的（在这一限制下，证明的步数显然为奇数步），搜索“仅”需要指数级的运行时间，否则按照课本上“证明”的定义对应的“最短”，我们需要考虑每个定理的所有可能的证明而不仅仅是最短证明，这至少需要指数的指数级的运算时间。

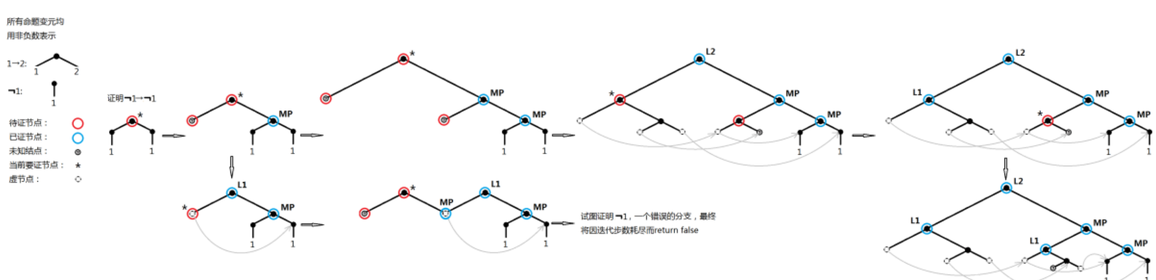
本程序采用自顶向下的 IDA*搜索算法（基于迭代加深的 A*算法）：

1. 初始待证明命题集 P 中只有目标命题 p0，迭代深度限制 bound 初始为 1，若 IDA*搜索结果为 false，bound=bound+2，再对 P={p0}进行 IDA*搜索直至搜索结果为 true；

2. 每步迭代对 P 中的一个命题 pi 依次匹配 L1、L2、L3，若其可能匹配成功，则对相应分支进行迭代（bound 减 1，从 P 中去掉 pi），迭代结果为 true 则输出证明并返回 true，否则匹配下一个公理；若三个公理均匹配失败或迭代返回 false，新建一个待定节点 pj 作为 MP 证明的前件，新建一个蕴含节点 pk，其左节点为 pj，右节点为 pi，从 P 中去掉 pi 并加入 pj、pk，bound 减 1，进行下一步迭代，若迭代结果为 false，返回 false，否则输出证明并返回 true；

3. 迭代终止条件为 bound 归 0 且 P 不为空（此时返回 false）或 P 为空（此时返回 true）。

另，一个证明过程的示意图（大图见打包文件夹中）：



首先证明：若程序能输出结果，其一定是 p0 的一个证明

每一步迭代证明的命题 pi，都是 L 型公理或由 MP 规则得到，对于待定节点，在匹配 L 型公理时，equal 函数会将其变为虚节点指向被匹配的节点，从而建立相等约束，因此匹配 L 型公理的命题即使内部有待定节点，未来对待定节点的修改不会影响其匹配 L 型公理的正确性，因此输出结果一定是 p0 的一个证明。

再证明：当输入一个可证明的命题时，程序可停机，并给出最短的证明

程序有三个递归函数 ida、equal、main：

ida 的参数 bound 每一次递归都会递减，归 0 时函数会结束递归，显然 ida 不会无限递归；

对 equal 函数，情况复杂一些，反证，假设 equal 会无限递归，由于节点的生成只在 ida 函数中发生，因此节点数有限，而 equal 函数比较两个节点 a、b 时，递归比较只发生在{a 及其后代}和{b 及其后代}之间，若产生无限递归，a 一定在一个环中，b 也同样（成环意味着一个命题内部的子命题包含自身，这会导致悖论），而这种情况仅可能在将一个节点设为另一个节点的后代时发生，因此在 equal 中将一个待定节点 a 变为虚节点指向被匹配的节点 b 时，需要先判断 a 不是 b 的后代，避免成环后，equal 就是可停机的；

对 main 中的循环，若命题可证，bound 递增，在有限时间内，程序总会找到一个证明，并且显然程序搜索了所有可能，bound 的递增足够紧，因此给出的证明还是最短的证明。

下面附上一些经典命题的证明结果：

否定肯定律((~p -> p)-> p)证明结果：

```
bound=3
bound=5
bound=7
bound=9
bound=11
bound=13
bound=15
bound=17
bound=19
0:      L1              ((~[1]->[1])->([-1]->(~[1]->[1])))
1:      L1              (~[1]->(~~([-1]->(~[1]->[1]))->~[1]))
2:      L3              ((~~([-1]->(~[1]->[1]))->~[1])->([1]->~([-1]->(~[1]->[1]))))
3:      L1              (((~~([-1]->(~[1]->[1]))->~[1])->([1]->~([-1]->(~[1]->[1]))))->(~[1]->((~~([-1]->(~[1]->[1]))->~[1])->([1]->~([-1]->(~[1]->[1]))))))
4:      2, 3, MP        (~[1]->((~~([-1]->(~[1]->[1]))->~[1])->([1]->~([-1]->(~[1]->[1])))))
5:      L2              ((~[1]->((~~([-1]->(~[1]->[1]))->~[1])->([1]->~([-1]->(~[1]->[1])))))->((~[1]->(~~([-1]->(~[1]->[1]))->~[1]))->(~[1]->([1]->~([-1]->(
~[1]->[1]))))))
6:      4, 5, MP        ((~[1]->(~~([-1]->(~[1]->[1]))->~[1]))->(~[1]->([1]->~([-1]->(~[1]->[1])))))
7:      1, 6, MP        (~[1]->([1]->~([-1]->(~[1]->[1]))))
8:      L2              ((~[1]->([1]->~([-1]->(~[1]->[1]))))->((~[1]->[1])->(~[1]->~([-1]->(~[1]->[1])))))
9:      7, 8, MP        ((~[1]->[1])->(~[1]->~([-1]->(~[1]->[1]))))
10:     L3              ((~[1]->~([-1]->(~[1]->[1])))->(([-1]->(~[1]->[1]))->[1]))
11:     L1              (((~[1]->~([-1]->(~[1]->[1])))->(([-1]->(~[1]->[1]))->[1]))->((~[1]->[1])->((~[1]->~([-1]->(~[1]->[1])))->(([-1]->(~[1]->[1]))->[1]))
))
12:     10, 11, MP      ((~[1]->[1])->((~[1]->~([-1]->(~[1]->[1])))->(([-1]->(~[1]->[1]))->[1])))
13:     L2              (((~[1]->[1])->((~[1]->~([-1]->(~[1]->[1])))->(([-1]->(~[1]->[1]))->[1])))->(((~[1]->[1])->(~[1]->~([-1]->(~[1]->[1]))))->((~[1]->[1]
)->(([-1]->(~[1]->[1]))->[1]))))
14:     12, 13, MP      (((~[1]->[1])->(~[1]->~([-1]->(~[1]->[1]))))->((~[1]->[1])->(([-1]->(~[1]->[1]))->[1])))
15:     9, 14, MP       ((~[1]->[1])->(([-1]->(~[1]->[1]))->[1]))
16:     L2              (((~[1]->[1])->(([-1]->(~[1]->[1]))->[1]))->(((~[1]->[1])->([-1]->(~[1]->[1])))->((~[1]->[1])->[1])))
17:     15, 16, MP      (((~[1]->[1])->([-1]->(~[1]->[1])))->((~[1]->[1])->[1]))
18:     0, 17, MP       ((~[1]->[1])->[1])
```

否定前件律(~q -> (q -> p))的证明：

```
bound=3
bound=5
bound=7
0:      L1              (~[2]->(~[1]->~[2]))
1:      L3              ((~[1]->~[2])->([2]->[1]))
2:      L1              (((~[1]->~[2])->([2]->[1]))->(~[2]->((~[1]->~[2])->([2]->[1]))))
3:      1, 2, MP        (~[2]->((~[1]->~[2])->([2]->[1])))
4:      L2              ((~[2]->((~[1]->~[2])->([2]->[1])))->((~[2]->(~[1]->~[2]))->(~[2]->([2]->[1]))))
5:      3, 4, MP        ((~[2]->(~[1]->~[2]))->(~[2]->([2]->[1])))
6:      0, 5, MP        (~[2]->([2]->[1]))
```

同一律(p -> p)的证明：

```
bound=3
bound=5
0:      L1              ([1]->([-1]->[1]))
1:      L1              ([1]->(([-1]->[1])->[1]))
2:      L2              (([1]->(([-1]->[1])->[1]))->(([1]->([-1]->[1]))->([1]->[1])))
3:      1, 2, MP        (([1]->([-1]->[1]))->([1]->[1]))
4:      0, 3, MP        ([1]->[1])
```


作者：

PB15030773

朱一铭