```
list1 = []


class SymbolLookupTable(object):
    '''A class implementing a Symbol Lookup Table that records each
   identifier's name, type, and scope.'''

    def __init__(self):
        '''Create a new empty table with the given name'''
        self.list = []

    def addEntry(self, entry):
        '''Add the given entry to the table.  Throws an error if
       there is already an entry with the given name and scope, regardless
       of the type.'''

        # throws error if a function is attempted to be declared
        # outside of the global block
        if entry.type == "function" and entry.scopeList[-1] != 0:
            return False
        # this will call an error in the Semantic Analyzer

        if self.verifyEntry(entry):
            return False

        # if not, add the entry to the table
        self.list.append(entry)
        return True

    def verifyEntry(self, entry):
        '''Verify that a given entry is in the table with the appropriate
       scope. Checks entry.validateWithTableEntry() on each
       entry in the table that has the same id as entry

       Returns true if entry exists in the table, regardless of type,
       meaning that addEntry should not work'''

        for x in self.list:
            if entry.validateWithTableEntry(x):
                return True

        # if none validate
        return False

    def getEntry(self, entryQuery):
        '''Returns the entry corresponding to the specified id and scope.'''

        for x in self.list:
            if entryQuery.validateWithTableEntry(x):
                return x

        return None


class SymbolTableEntry:
    '''A class representing a SymbolLookupTable entry. Each entry has
   an id (name), maybe a type, scope list, function string (to indicate if
   the entry is a part of a function), and function parameter types (if a function).'''

    def __init__(self):
        '''Default constructor, initializes everything to
       empty'''
        self.id = ""
        self.type = ""
```

```
        self.scopeList = []
        self.function = None
        self.functionParameterTypes = []
        self.initialized = False
        self.functionParamBool = False

    def __init__(self, inId, inType, inScopeList, inFunction, inFunctionPTypes):
        '''Sets the entry's id, type, scope list, function string,
       and function parameter type list'''
        self.id = inId
        self.type = inType
        self.scopeList = inScopeList
        self.function = inFunction
        if inFunctionPTypes != None:
            self.functionParameterTypes = list(inFunctionPTypes)
        else:
            self.functionParameterTypes = None
        self.initialized = False
        self.functionParamBool = False

    def validateWithTableEntry(self, tableEntry):
        '''Returns true if the existence of tableEntry means that
       self cannot be added to the table (same ID and overlapping scopes)
       Ignore type since we don't want to allow different types'''
        idEq = (self.id == tableEntry.id)
        topScopeCountTableEntry = tableEntry.scopeList[-1]
        selfScopeAcceptable = topScopeCountTableEntry in self.scopeList
        # if this is a function, it can be used anywhere
        if self.type == "function":
            functionScopeAcceptable = True
        # otherwise, check to make sure we are using variables in the right function
        # or in a non-function scope
        else:
            functionScopeAcceptable = (self.function == tableEntry.function)
        if self.function != None and idEq and functionScopeAcceptable and
tableEntry.functionParamBool == True:
            return True
        elif idEq and selfScopeAcceptable and functionScopeAcceptable:
            return True
        else:
            return False
```