CENG 242
Homework # 4

(Due: April 16th, 2006 Sunday 23:59)

Hypothetical version of C described in Hw #3 is back. This time you will implement a class named SymTable in C++.

```
struct Error {
      string symbol;
      int lineno;
};

class SymTable {
  public:
      SymTable( ) ;           // constructs an empty table
      SymTable(const SymTable & rhs) ;    // copy constructor
      ~SymTable( ) ;          // destructs the table

      void declVar(string symbol) ;       // adds symbol to current
                  // scope in table. The address of the symbol is the
                  // current line number. You should throw Error typed
                  // value including the name of the symbol and the
                  // line number of the previous definition when there
                  // exist a symbol with the same name in the same
                  // scope.
      void declFunc(string symbol) ;      // adds symbol to current
                  // scope in table and creates a new local scope for
                  // the new function. You should throw Error typed
                  // value including the name of the symbol and the
                  // line number of the previous definition when there
                  // exist a symbol with the same name in the same
                  // scope.
      void exitFunc( ) ;      // terminates the last scope in the table
      void refIdentifier(string symbol) ; //adds a binding to the table
      SymTable link(SymTable t2, int offset) ;  // merges the table
                  // with t2 and returns the new table. Unresolved
                  // symbols of one table can be bound to declarations
                  // of the other and the resulting table is returned
                  // with all binding and applied occurrences are
                  // combined. Offset is added to all line numbers of
                  // second table, so line numbers are not confused.
                  // You should throw Error typed value including the
                  // name of the symbol and the line number of the
                  // definition in the first (current) table when a
                  // symbol in the first table also exists in the
                  // second table in the same scope. When there are
                  // more than one redefinition, give the identifier
                  // with the smallest line number in first table.

      SymTable & operator=(const SymTable &) ; // operator =
      int operator[](string symbol) ; // returns the line number of the
                  // definition of the symbol, and if not found returns
                  // -1
```

```
      friend ostream & operator<<(ostream & ost, const SymTable & t) ;
                  // prints the symbol table
}
```

Specifications:

- You should not add any public member to class.
- In error conditions mentioned above, only throw the exception. Do not write any code to catch it in your submission.
- You should not use any **STL** classes except **string**.
- Use **cow** submission system.
- Put your declaration into a file named “SymTable.h” and definition to a file named “SymTable.cpp”. **tar** and submit these two files. You should not have **main** function or any other global function.
- Your code should be compiled on machines in **mera** labs with the compiler **g++**. If not compiled, you will get 0.