Birla Institute of Technology & Science, Pilani – K K Birla Goa Campus
First Semester: 2017-2018
Assignments

September 1, 2017
***Course No.*: CS F301** ***Course Title*: Principles of Programming Languages**
***Instructor-in-Charge*: Prof. Santonu Sarkar** ***Instructor*: Gargi Alavani**
***E-mails*:** `santonus@goa.bits-pilani.ac.in;` `p2016008@goa.bits-pilani.ac.in`

Preface

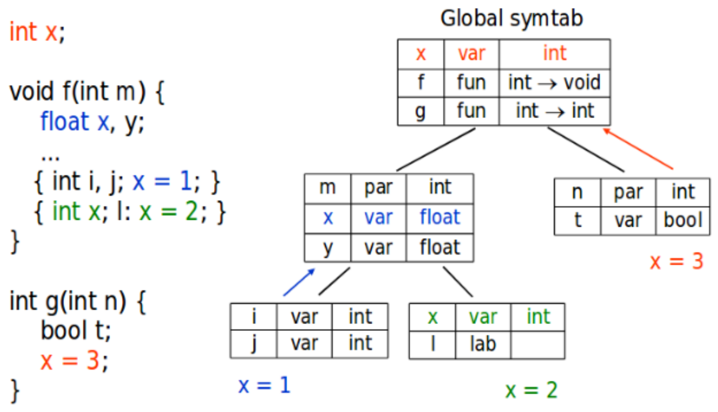
The assignments given here are meant to improve your knowledge on programming language, concepts and programming styles. In the Internet era we live in, it is impossible that you will NOT find these assignments and their answers or their variants somewhere. So it is very easy for you to copy them and show it to us. You can also copy the solution from your friend. Please note that I have no time or interest to check who is taking such a shortcut or who is genuinely trying. Remember, if you take a short-cut, it won't lead you anywhere; you will only deceive yourself, not me.

1 Symbol Table

In this assignment you are expected to design a symbol table structure for the semantic analysis stage of the compiler using Java/C++.

The Symbol table structure is implemented using a heirarchy of scopes. It can be efficiently implemented using tree structure as shown in fig . You add symbol table entries to a nested new node when you enter a new scope and delete entries when you exit scope. You lookup up symbol table information by searching for an identifier in a bottom up fashion.

For every entry in the symbol table you store <identifier,type,kind>
for eg. <sum,func,int> , <a,var,int> , <c,var,char>



Define following operations for the symbol table:
insert(name,kind,type): inserts an entry for x in the current scope if it is already not defined in it. Throws an error if the variable name is already present in the most recent scope.
lookup(name): returns the most recent definition of name by searching the tree from leaf to root. If no match is found it throws an error.
enter_scope(): Generates a new level of nesting by creating a symbol table for the new scope.
exit_scope(): remove symbol table entries for the current scope and move back to the enclosing scope in the symbol table tree.

Table 1: Node I

f	function	void
g	function	void

Table 2: Node II

a	parameter	int
b	parameter	int
x	variable	double
...	...	...

You are expected to use a struct/class for a symbol table entry with members as <identifier,type,kind>
.

Each node of the symbol table tree has:

- A Symbol table(collection of symbol table entries)
- Pointer to a parent node
- Collection of pointers to children nodes.

You may assume that the types are restricted to int,float,double,bool and the constructs belong to the set variable,function,parameter. Also assume that functions can have up to two parameters. You are supposed to implement a static scope for this assignment, Depending on your implementation you may have to store pointer to the symbol table of the calling function while implementing the function exit_scope().

Once you construct symbol table, test how symbol table changes/grows in processing the following code:

```
void f(int a, int b)
{
double x;
int i=0;
while (i<5)
{
int y;
}
}
void g() {
int x;
f();
}
```

Generated symbol tree will have nodes such as 1 and 2.

Test this table by providing following input to observe the errors thrown by insert and lookup functions.

```
Void f()
{
   double x;
}
void g()
{
 float x;
}
```

Since regular expressions were covered in the tutorials, you are recommended to use them to parse the input. This is a challenging yet doable task. However, considering the complexity of this implementation, using the tuples<identifier,type,kind> as input is also acceptable. Work out all the tuples for given input code and take it as input tuples for constructing the symbol table. Print the generated symbol table. In case you are opting for option A, parse the input .c file by using specific regular expresssions. You are not expected to design a generic parser.

Instructor-In-Charge, CS F301
Santonu Sarkar