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

September 30, 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

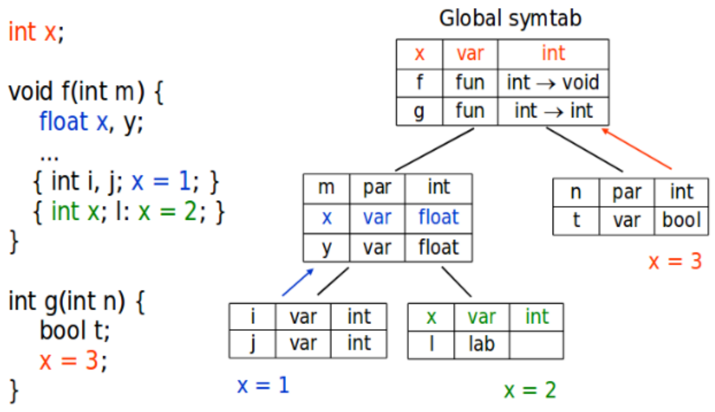
The assignments given here are meant to improve your knowledge on programming language, concepts and programming styles. In the Internet era we live in, it is impossible that you will NOT find these assignments and their answers or their variants somewhere. So it is very easy for you to copy them and 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().

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.

2 Browser Network Tasks

A browser tab or download could be composed of several network activity tasks. For example, you may need to download several page elements for loading a website, and each of the elements could be an independent network task. Similarly a file download could be run in several parts, to be assembled together into a single file later.

You could run each of the network tasks as a separate thread and listen for their completion in a main thread to keep track of how much of the download/loading has finished.

For this assignment, you need to have the following classes:

- Tab: A tab loads a webpage which can create multiple Tasks equal to the number of separate web elements that need to be downloaded to display on the webpage.
- Download: A single download which can create multiple Tasks equal to the number of fragments you are dividing the file into.
- Task: This implements runnable, and represents a network activity task. Every task has some size (say, 200), which is a representative of how much time it would take to complete(in this case, 200 miliseconds). Each task runs in its own separate thread.

For every Tab, while creating its tasks assign them random sizes in a range (for eg. 100 to 1000) and keep track of the total size of all its tasks. Every time a task thread completes, it should signal an event, and when its source (tab/download) receives that event, the event handler should print out the percentage of total size loaded/completed.

When you execute the following in main() :

```
Tab t1=new Tab(10, "google.com");
```

Note : *10 is the number of tasks the page is suppposed to load. Assume we know it when creating the tab. Also, we necessarily provide it with a url to load*/

Output should print:

```
Total size of tasks=416
google.com Loaded 1%.
google.com Loaded 4%.
google.com Loaded 12%.
google.com Loaded 19%.
google.com Loaded 27%.
google.com Loaded 36%.
google.com Loaded 46%.
google.com Loaded 61%.
google.com Loaded 78%.
google.com Loaded 100%. Tab idle.
```

Note here that just calling the Tab() constructor started the task threads. You will also have to implement an Interface for listeners and an event class for task completion event. Take care of concurrency issues while keeping trak of how many tasks have finished. At the end, we expect your program to work something like this:

In main(), when you execute the following:

```
Tab t1=new Tab(10, "google.com");
Tab t2=new Tab(7, "amazon.com");
```

Output should be something similar to:

```
Total size of tasks=580
Total size of tasks=206
amazon.com Loaded 1%.
amazon.com Loaded 7%.
amazon.com Loaded 14%.
google.com Loaded 2%.
amazon.com Loaded 24%.
amazon.com Loaded 36%.
google.com Loaded 7%.
google.com Loaded 13%.
google.com Loaded 20%.
google.com Loaded 29%.
google.com Loaded 40%.
amazon.com Loaded 68%.
amazon.com Loaded 100%. Tab idle.
google.com Loaded 51%.
google.com Loaded 66%.
google.com Loaded 83%.
google.com Loaded 100%. Tab idle.
```

Assuming you are properly running the threads, your program is correct if the page/download loads till 100% everytime (means that all events are being received and accounted). If not, check for modification/reading data members in unsafe state.

Instructor-In-Charge, CS F301
Santonu Sarkar