Ahsanullah University of Science and Technology
Department of Computer Science and Engineering
Examination: Assignment 2 Session: Fall 2022
Year/Semester: 4/1 Course No: CSE4130
Course Title: Formal Languages and Compilers Lab

ASSIGNMENT #3:

Suppose, a given C source program has been scanned, filtered and then lexically analyzed as it was done in Session 1 & 2. We have all the lexemes marked as different types of tokens like keywords, identifiers, operators, separators, parentheses, numbers, etc. Now we generate a Symbol Table describing the features of the identifiers. Then, we generate a modified token stream in accordance with the Symbol Table for processing by the next phase, that is, Syntax Analysis.

Sample source program:

```
// A program fragment
float x1 = 3.125;
/* Definition of the
function f1 */
double  f1(int x)
{
  double z;
  z = 0.01;
  return z;
}
//* Beginning of 'main'
int main(void)
{
  int n1;  double z;
  n1=25;  z=f1(n1);
```

Sample input based on the program fragment:

[kw float] [id x1] [op =] [num 3.125] [sep ;] [kw double] [id f1] [par (] [kw int] [id x] [par)] [brc {] [kw double] [id z] [sep ;] [id z] [op =] [num 0.01] [sep ;] [kw return] [id z] [sep ;] [brc }] [kw int] [id main] [par (] [kw void] [par)] [brc {] [kw int] [id n1] [sep ;] [kw double] [id z] [sep ;] [id n1] [op =] [num 25] [sep ;] [id z] [op =] [id f1] [par (] [id n1] [par)] [sep ;]

Sample input based on the program fragment:

[kw float] [id x1] [op =] [num 3.125] [sep ;] [kw double] [id f1] [par (] [kw int] [id x] [par)] [brc {] [kw double] [id z] [sep ;] [id z] [op =] [num 0.01] [sep ;] [kw return] [id z] [sep ;] [brc }] [kw int] [id main] [par (] [kw void] [par)] [brc {] [kw int] [id n1] [sep ;] [kw double] [id z] [sep ;] [id n1] [op =] [num 25] [sep ;] [id z] [op =] [id f1] [par (] [id n1] [par)] [sep ;]

Step 1: After complete recognition of all the lexemes only identifiers are kept in pairs for formation of Symbol Tables. The token stream should look like the one as follows:

[float] [id x1] [=] [3.125] [;] [double] [id f1] [(] [int] [id x] [)] [{] [double] [id z] [;] [id z] [=] [0.01] [;] [return] [id z] [;] [}] [int] [id main] [(] [void] [)] [{] [int] [id n1] [;] [double] [id z] [;] [id n1] [=] [25] [;] [id z] [=] [id f1] [(] [id n1] [)] [;]

Step 2: Symbol Table generation:

Symbol Table:

Sl. No.	Name	Id Type	Data Type	Scope	Value
1	x1	var	float	global	3.125
2	f1	func	double	global	
3	x	var	int	f1	
4	z	var	double	f1	0.01
5	main	func	int	global	
6	n1	var	int	main	25
7	z	var	double	main	

Sample source program:

```
// A program fragment
float x1 = 3.125;
/* Definition of the
function f1 */
double  f1(int x)
{
   double z;
   z = 0.01;
   return z;
}
//* Beginning of 'main'
int main(void)
{
  int n1;  double z;
  n1=25;  z=f1(n1);
```

Step 3: Your program should implement the following functions on the symbol table.

1. **insert()** : *Insert a name in the symbol table*
2. **set_attribute()** : *associate an attribute with an existing entry*
3. **free()** : *remove all entries and free storage of symbol table(user choice)*
4. **lookup()** : *search for a name and return to the pointer (can be serial no) from the table*
5. **display()** : *display the whole symbol table(user choice)*

Step 4: Modified token stream for Syntax Analysis:

Sample source program:

```
// A program fragment
float x1 = 3.125;
/* Definition of the
function f1 */
double  f1(int x)
{
  double z;
  z = 0.01;
  return z;
}
//* Beginning of 'main'
int main(void)
{
int n1;  double z;
 n1=25;  z=f1(n1);
```

[float] [id 1] [=] [3.125] [;] [double] [id 2] [(] [int] [id 3] [)] [{]
[double] [id 4] [;] [id 4] [=] [0.01] [;] [return] [id 4] [;] [}] [int] [id 5]
[(] [void] [)] [{] [int] [id 6] [;] [double] [id 7] [;] [id 6] [=] [25] [;] [id
7] [=] [id 2] [(] [id 6] [)] [;]