CSE 3104: Compiler Sessional

Assignment 01
Construction of a Symbol-Table

We will build a symbol-table in this assignment. At this initial stage of the project, we will omit many details regarding an actual symbol-table and we will simply adhere to the basic concept that “a symbol-table is an efficient data-dictionary for the symbols used in a program". Thus, our focus in this assignment is to construct a simple hash-based data-dictionary.

Input:
The input to your program will be a sequence of two-tuples, where each element in each tuple is a string. An example of input sequence is given below.

- int, INTEGER
- myFunction, FUNCTION
- x, IDENTIFIER
- 5, NUMBER

The first element of each tuple will be the key of the record to be stored in the symboltable. Hence, you have to apply the hash function on the first element of each tuple.

Implementation Issues:

Implement the following two classes:

o class SymbolInfo: The definition of this class will grow gradually throughout the development of this project. For this assignment, we simply need two members, one for storing the symbol (e.g. “x”) and another for storing the type of the symbol (e.g. “IDENTIFIER”).

o **class SymbolTable:** Since our symbol-table will be a hash-table based on chaining, we will have to start with an array of pointers where each pointer points to a list of nodes of type class SymbolInfo. class SymbolTable will have such an array of pointers. For this assignment, the choice of the size of this array, as well as of the hash function is left upto you(20 can be preferrable). In addition to this array of pointers, class SymbolTable will have three functions for the following purposes.

- insert(): to insert a new symbol along with its type into the symbol table.
- lookup(): to lookup whether a given symbol already exists in the symbol table or not
- dump(): to dump the contents of the symbol table to the console.
- erase(): to delete a certain symbol

Some Other Instructions:

- Do not copy. Plagiarism will cause 100% marks deduction for both source and target.
- Tips: To avoid matching your code with someone else, you can use different types of hashing method for inserting.

Sample Skeleton

```
class SymbolInfo            // class for storing a symbol and
its type with a next pointer
{
public:
     string symbol;
     string type;
     SymbolInfo* next;
};
int askey(string s){
    //function to convert a string to some number
    //you can convert the string characters to
    their ascii values and sum them
    //or you can use any other method to convert
    a string to a corresponding number value.
    // this value will be used for hashing
}
class SymbolTable           //class creating hashtable ,
functions to insert,search,dump,delete symbols
{
    public:
         int i;
         SymbolInfo* hash;
         SymbolTable()//creating a blank hashtable
         {
             Hash = new SymbolInfo[TABLESIZE];
             for(i=0;i<TABLESIZE;i++)
             {
                 hash[i].next=NULL;
             }
         }
        void insert()       // function to insert new symbols in
hash table
        {
```

```
          cout<<"\nInput symbol and type to insert:  \n";
          cin>>head->symbol>>head->type;
          int in=askey(head->symbol)%TABLESIZE;
        }

        void dump()     //function to print the whole hashtable
        {
        }
       void lookup() {      //function to search desired symbol
        {

        void erase()    // function to delete a certain input
symbol
        {
        }
};


void menu()
{
    cout<<"\n*Press 1 to insert a new symbol along with its type
into the symbol-table\n";
    cout<<"\n*Press 2 to...";
    cout<<"\n*Press 3 to...";
    cout<<"\n*Press 4 to...";
    cout<<"\n*Press 0 to exit\n";
}

int main()
{
    int choice;
    SymbolTable x;
    menu();
    cin>>choice;
    while(choice!=0)
    {
        switch(choice){
            //insert/lookup/dump/erase based on choice value
        }
        menu();
        cin>>choice;
    }
    return 0;
}
```

Sample Output:

Option Menu:

```
*Press 1 to insert a new symbol along with its type into the symbol-table

*Press 2 to lookup whether a given symbol already exists in the symbol-table or not

*Press 3 to dump the contents of the symbol table to the console

*Press 4 to delete a given symbol if it already exists in the symbol-table

*Press 0 to exit
```

Pressed 1:

```
1

Input symbol and type to insert:
myFunction
FUNCTION
```

Pressed 3:

```
3

0 : x ( IDENTIFIER ) , i ( INTEGER )
1 :
2 :
3 : myFunction ( FUNCTION )
4 :
5 :
6 :
7 :
8 :
9 :
10 :
11 : ) ( PARENTHESIS )
12 : f ( FUNCTION )
13 :
14 :
```

Pressed 2:

```
2

Input symbol to lookup :
x
```

Pressed 4:

```
4

Input symbol to delete :
i
```

Prepared By:

Faria Tabassum

Lecturer, Department of CSE

[redacted]