CS335: Milestone 3

Group 14

Members:

Name	Roll No.	Email
Supreeth Baliga	180801	supbal@iitk.ac.in
Chinmay Goyal	180206	chinmayg@iitk.ac.in
Aaryan Srivastava	180007	aaryans@iitk.ac.in
Nikhil Agarwal	180475	nikhilag@iitk.ac.in
Sanchit Agrawal	180664	sanagrwl@iitk.ac.in

Source Language (S): *C*
Implementation Language (I): *Python3*
Target Language (T): *X86 Assembly*

Repo Link: https://github.com/SupreethBaliga/Compilers/

```
TOP
├── src/
│   ├── lexer.py (The older version of lexer)
│   ├── lexerClass.py (The class oriented version of lexer)
│   ├── parser.py (Creates the AST for Milestone 2)
│   ├── parserClass.py (Creates the AST and does semantic analysis for Milestone 3)
│   ├── SymbolTable.py (Handles all functionality related to symbol table)
│   └── TypeTable.py (Handles all functionality related to type table)
├── tests/
│   └── (Contains all tests for respective milestones)
├── PDFs
│   └── (Contains the corresponding PDF files for milestones)
├── dot
│   └── (Contains the dot files corresponding to the input given)
├── ASTgraphs
│   └── (Contains the graphical view of the constructed ASTs of the input codes)
├── ST
│   └── (Contains all the dumped symbol tables of the input codes)
├── tmp
│   └── (Contains temporary files created by the parser if -d flag is set)
├── requirements.txt
├── install.sh
├── lex.sh
├── parser.sh
├── README.md
└── .gitignore
```

Instructions to run:

- Install all the python modules and set up virtual environment by executing *$ install.sh*
- Activate the virtual environment by running: *$ source venvcompiler/bin/activate*

- To use the parser execute *$ ./parser.sh* and provide the test file(s) as command line arguments

Output of *'$ git show parser '*:

```
commit c9c99fa61810a884796f50a94ab749422f4189be (HEAD -> main, tag: semantics, origin/main,
origin/HEAD)
Author: Roberticey <48215918+Roberticey@users.noreply.github.com>
Date:   Thu Apr 8 19:28:54 2021 +0530

    Update parserClass.py

diff --git a/src/parserClass.py b/src/parserClass.py
index 03df41d..202d733 100644
--- a/src/parserClass.py
+++ b/src/parserClass.py
@@ -165,7 +165,6 @@ class CParser():
         if found: # Change this accordingly



-
             try :
                 entry['type']
             except:
@@ -1590,6 +1589,10 @@ class CParser():
                 print(f'Struct / Union type variable not allowed as first operand of
                 ternary operator')
                 return

+            elif p[3] == None or p[5]==None:
+                self.ST.error = 1;
+                print(f'Cannot perform conditional operation at line {p.lineno(2)}')
+                return

             elif p[3].type in [None, []] or p[5].type in [None, []] :
                 self.ST.error = 1;
@@ -3295,8 +3298,8 @@ class CParser():
                         self.ST.ModifySymbol(var_name, "vars", found['vars'],
                         p.lineno(0))
                         self.ST.ModifySymbol(var_name, "check", found['check'],
                         p.lineno(0))
                         self.ST.ModifySymbol(var_name, "type", p[0].variables[var_name],
                         p.lineno(0))
-                    else:
-                        self.ST.ModifySymbol(var_name, "type", p[0].variables[var_name],
p.lineno(0))
+                else:
+                    self.ST.ModifySymbol(var_name, "type", p[0].variables[var_name],p.lineno(0))
                 self.ST.ModifySymbol(var_name, "check", "PARAM", p.lineno(0))
                 param_nums += 1
```

Points to Note:

- We needed to change our lexer file to align it with the grammar rules.
- The AST now only contains executable operations. The declarations are inserted into the symbol table.
- The ANSI standard link that was provided was not in accordance with the modern C standard. Some changes in the grammar needed to be made.
  - The ANSI standard link only allowed for declarations to be done first and then all the statements. We modified it so that the program can now have mixture of declarations and statements without any positional restriction enforced by ANSI-C.

  - Similarly, the ANSI-C grammar does not allow us to declare variables within the initialization statement of a for loop. We modified it accordingly.

- Tests for Milestone-3 have been provided in a different directory (tests/Milestone3/).
- A bash script parser.sh is provided to run the parser. To check the available flags, run parser.sh with -h[–help].
- The grammar needed to be modified and marker symbols were inserted in some rules in order to inherit information, do proper scope management in symbol table and also to input parameter information for the function in the symbol table.
- The milestone documentation asked us to print the Symbol Table in .csv format. But due to some constraints, we spoke to sir and he allowed us to print the symbol tables in the .txt format.