CSC467 HW1

Team Members:
Yiming Kang 998676730
Zexuan Wang 998851773

How we dealt with special problems encountered: 1- Deterministic and systematic testing: The main problem we encountered was the lack of good testing frameworks. Manually checking over -Tn output is both labor intensive and inaccurate. Therefore we came up with our own testing framework

- diff target -Tn output with -Tn output of each run
- check the consistency of tokens generated within a file (keyword int should have the same token)

```
#!/bin/bash
echo "==================== TESTING STARTS ===================="
echo ""
for file in ./*.frag; do
    ../compiler467 -Tn "$file" > "$file".out 2>&1
    DIFF=$(diff "$file".out "$file".target)
    OUT=$(python verify.py "$file".out)
    if [ "$DIFF" != "" ]
    then
        printf "diff   %-30s --- \e[1;31mFAILED\e[1;0m\n" "$file"
        echo "*****TEST $file FAILED TO MATCH TARGET*****"
        diff "$file".out "$file".target
    else
        printf "diff   %-30s --- \e[1;32mOK\e[1;0m\n" "$file"
    fi

    if [ "$OUT" != "" ]
    then
        printf "verify %-30s --- \e[1;31mFAILED\e[1;0m\n" "$file"
        echo "*****TEST $file FAILED TO PRODUCE CONSITENT TOKEN*****"
        python verify.py "$file".out
    else
        printf "verify %-30s --- \e[1;32mOK\e[1;0m\n" "$file"
    fi
done
rm -f *.out
echo "==================== TESTING ENDS ====================="
```

```
import re
import sys

KNOWN_TOKENS = [
        'if', 'else', 'for', 'while', 'void', ',', '(', ')',
        '{', '}', '[', ']', ':', 'bool', 'int', 'float',
        '"', 'dp3', 'lit', 'rsq', '=', 'const', ';', '\'',
]

def main(file_name):
    lex_output_regex = re.compile(r"TOKEN\s+(\d{3})\s+:\s+(.*)")
    seen_tokens = dict()
    ret = 0
    line_num = 0
    with open(file_name, 'r') as fin:
        for line in fin:
```

```
            line_num += 1
            matched = lex_output_regex.match(line)
            if matched is None:
                continue
            value = matched.group(1)
            token = matched.group(2)
            if token in KNOWN_TOKENS:
                if token in seen_tokens.keys():
                    if seen_tokens[token] != value:
                        print "ERROR: Token ", token, " is asigned a new value on line #", str(line_num
                        ret = -1
                else:
                    seen_tokens[token] = value
    # print seen_tokens
    return ret

if __name__ == '__main__':
    if len(sys.argv) != 2:
        print "ERROR: Invalid argument"
        exit(0)
    if main(sys.argv[1]) == -1:
        print "***** TEST FAILED *****"
```