Homework 4

Programming Language Concepts

Due April 10, 2020

Name: Harry Park

1. (points) Consider the following skeletal C program:

```
void fun1(void); /* prototype */
void fun2(void); /* prototype */
void fun3(void); /* prototype */

void main() {
   int a, b, c;
   . . .
}
void fun1(void) {
   int b, c, d;
   . . .
}
void fun2(void) {
   int c, d, e;
   . . .
}
void fun3(void) {
   int d, e, f;
   . . .
}
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last function called? Include with each visible variable the name of the function in which it was defined.

a. main calls fun1; fun1 calls fun2; fun2 calls fun3.

b. main calls fun1; fun1 calls fun3.

c. main calls fun2; fun2 calls fun3; fun3 calls fun1.

d. main calls fun3; fun3 calls fun1.

e. main calls fun1; fun1 calls fun3; fun3 calls fun2.

f. main calls fun3; fun3 calls fun2; fun2 calls fun1.

a. a (defined in main)
b (defined in fun1)
c (defined in fun2)
d, e, f (defined in fun3)

b. a (defined in main)
b, c (defined in fun1)
d, e, f (defined in fun3)

c. a (defined in main)
e, f (defined in fun3)
b, c, d (defined in fun1)

d. a (defined in main)
e, f (defined in fun3)
b, c, d (defined in fun1)

e. a (defined in main)
b (defined in fun1)
f (defined in fun3)
c, d, e (defined in fun2)

f. a (defined in main)
f (defined in fun3)
e (defined in fun2)
b, c, d (defined in fun1)

2. (points) Write a JavaScript script that has subprograms nested three deep and in which each nested subprogram references variables defined in all of its enclosing subprograms.

3. (points) Repeat exercise 2 with Python.

4. (points) Write an EBNF rule that describes the while statement of Java or C++. Write the recursive-descent subprogram in Java or C++ for this rule.

2.

```
function main() {
    var x = 5;
    console.log(x);

    function sub1() {
        var y = x + 10;
        console.log(y);

        function sub2() {
            var z = (y + x) * 2;
            console.log(z);

            function sub3() {
                var sum = x + y + z;
                console.log(sum);
                console.log(' x = '+ x + '\n y = ' + y +
                        '\n z = '+ z +  '\n sum = ' + sum);
            }
            sub3();
        }
        sub2();
    }
    sub1();
}
main();
```

3.

```
def main():
    x = 5;
    print(x);

    def sub1():
        y = x + 10;
        print(y);

        def sub2():
            z = (y+x) * 2;
            print(z);

            def sub3():
                sum = x + y + z;
                print(' x = '+ str(x) + '\n y = ' + str(y) +
                    '\n z = '+ str(z) +  '\n sum = ' + str(sum));
            sub3()

        sub2()

    sub1()

main()
```


_____ 0 points

4.

<while_stmt> -> WHILE ‘(‘ (<arith_expr> | <logic_expr>) ‘)’

<block>

<block> -> <stmt> | ‘{‘ <stmt> {<stmt>} ‘}’

```
public enum Parser {
PROGRAM {
  void parse(String s) throws ParseException {
    s = s.trim();
    if (s.startsWith("begin") && s.endsWith("end")) {
      STATEMENT_LIST.parse(s.substring(5, s.length() - 3));
    } else {
      throw new ParseException("Illegal begin/end");
    }
  }
},

BLOCK {
  void parse(String s) throws ParseException {
    String[] parts = s.trim().split(";");
    for (String part : parts) {
      STATEMENT.parse(part.trim());
    }
  }
},

STATEMENT {
  void parse(String s) throws ParseException {
    if (s.startsWith("while")) {
      WHILE.parse(s.substring(5));
    } else if (s.contains("=")) {
      ASSIGNMENT.parse(s);
    } else {
      throw new ParseException("Illegal statement: " + s);
    }
  }
},

WHILE {
  void parse(String s) throws ParseException {
    int i = s.indexOf("(");
    int j = s.indexOf(")");
    if (i != -1 && j != -1) {
      LOGICAL.parse(s.substring(i + 1, j).trim());
      STATEMENT.parse(s.substring(j + 1).trim());
    } else {
      throw new ParseException("Illegal while: " + s);
    }
  }
},

EXPRESSION {
  void parse(String s) throws ParseException {
    String[] parts = s.split("\\+|-");
    for (String part : parts) {
      VARIABLE.parse(part.trim());
    }
  }
},

LOGICAL {
  void parse(String s) throws ParseException {
    int i;
    if (s.contains("<")) {
      i = s.indexOf("<");
    } else if (s.contains(">")) {
      i = s.indexOf(">");
    } else {
      throw new ParseException("Illegal logical: " + s);
    }
    VARIABLE.parse(s.substring(0, i).trim());
    VARIABLE.parse(s.substring(i + 1).trim());
  }
},

VARIABLE {
  void parse(String s) throws ParseException {
    if (!s.matches("^[a-zA-Z][a-zA-Z0-9]*$")) {
      throw new ParseException("Illegal variable: " + s);
    }
  }
};

abstract void parse(String s) throws ParseException;

class ParseException extends Exception {
        public ParseException(String message) {
         super(message);
        }
}
```