Name: Minh Binh Nguyen
PantherID: 002-46-4288

Programming Language Concept
TEST 2

Question 1:

Main.java:

```
// Question 1
// Programmer: Minh Nguyen

import java.io.File;
import java.io.FileNotFoundException;
import java.net.URL;
import java.util.HashMap;
import java.util.Scanner;
import java.util.Set;
import java.util.regex.Pattern;

public class Question1 {

    // Patterns
    static HashMap<String, String> regex = new HashMap<>();

    public static void initializeRegex() {
        regex.put("PERL_IDEN_SCALAR", "^(\\$)(_)?[a-zA-Z0-9]+_?[a-zA-Z0-
9]+");
        regex.put("PERL_IDEN_ARRAY", "^(@)(_)?[a-zA-Z0-9]+_?[a-zA-Z0-9]+");
        regex.put("PERL_IDEN_HASH", "^(%)(_)?[a-zA-Z0-9]+_?[a-zA-Z0-9]+");

        regex.put("JAVA_STRING", "^(\").*(\"$)");
        regex.put("C_INT_DECIMAL", "^[\\d]+");
        regex.put("C_INT_OCT", "^0[0-7]*");
        regex.put("C_INT_HEX", "^0[xX][0-9a-fA-F]*");
        regex.put("C_CHAR", "^\\'[a-zA-Z0-9]\\'");
        regex.put("C_FLOAT", "^[-+]?(\\d)+\\.(\\d)*([eE][-+]?\\d+)?");

        regex.put("STRING", "^String");
        regex.put("INTEGER", "^Integer");
        regex.put("CHARACTER", "^Character");
        regex.put("FLOAT", "^Float");
        regex.put("VOID", "^Void");

        regex.put("ADDITION", "^\\+");
        regex.put("ASSIGNMENT", "^=");
        regex.put("SUBTRACTION", "^-");
        regex.put("DIVISION", "^/");
        regex.put("MULTIPLICATION", "^\\*");
        regex.put("MODULO", "^%");
        regex.put("AND", "^\\&\\&");
        regex.put("OR", "^\\|\\|");
        regex.put("NOT", "^!");
```

```
        regex.put("OPEN_BLOCK", "^\\{");
        regex.put("CLOSE_BLOCK", "^\\}");
        regex.put("OPEN_FUNCTION", "^\\(");
        regex.put("CLOSE_FUNCTION", "^\\)");
        regex.put("OPEN_ARRAY", "^\\[");
        regex.put("CLOSE_ARRAY", "^\\]");
    }

    public static String readFile (String fileName) {

        String file = "";
        try {
            File myObj = new File(fileName);
            Scanner myReader = new Scanner(myObj);
            while (myReader.hasNextLine()) {
                String data = myReader.nextLine();
                file += data + "\n";
            }
            myReader.close();
        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
        return file;
    }

    public static void lex(String str) {

        Set<String> keys = regex.keySet();
        String output = "Lex " + str + " is ";
        boolean found = false;

        for (String k : keys) {
            String pattern = regex.get(k);
            if (Pattern.compile(pattern).matcher(str).find()) {
                found = true;
                output += k;
                if (k.equals("PERL_IDEN_ARRAY") || k.equals("PERL_IDEN_HASH")
|| k.equals("PERL_IDEN_SCALAR")){
                    if (str.charAt(1) == '_') {
                        output += " PRIVATE";
                    }
                }
                break;
            }
        }
        if (!found) {
            output += "UNKNOWN";
        }
        System.out.println(output);
    }

    public static void main(String[] args) {

        // Read file
        URL url = Question1.class.getResource("question1.in");
        String data = readFile(url.getPath());
```

```
        initializeRegex();
        // Split the string by spaces
        String listStrs [] = data.split("[ \\r\\n]");

        for (String str : listStrs) {
            lex(str);
        }
    }

}
```

question1.in:

```
@variable123 = 123 + 456 % 345
$_1234abc = 123.456 - 0x324 * 1234 / 5
$_abc_123 = "string"
%ABCD1234abc
'd'
'cd'
String
Integer
Character
Float
Void
{ $bool && $bool2 || $bool3 }
( )
@array1 [ ]
```

Output:

```
Lex @variable123 is PERL_IDEN_ARRAY
Lex = is ASSIGNMENT
Lex 123 is C_INT_DECIMAL
Lex + is ADDITION
Lex 456 is C_INT_DECIMAL
Lex % is MODULO
Lex 345 is C_INT_DECIMAL
Lex $_1234abc is PERL_IDEN_SCALAR PRIVATE
Lex = is ASSIGNMENT
Lex 123.456 is C_INT_DECIMAL
Lex - is SUBTRACTION
Lex 0x324 is C_INT_HEX
Lex * is MULTIPLICATION
Lex 1234 is C_INT_DECIMAL
Lex / is DIVISION
Lex 5 is C_INT_DECIMAL
Lex $_abc_123 is PERL_IDEN_SCALAR PRIVATE
Lex = is ASSIGNMENT
Lex "string" is JAVA_STRING
Lex %ABCD1234abc is PERL_IDEN_HASH
Lex 'd' is C_CHAR
Lex 'cd' is UNKNOWN
Lex String is STRING
Lex Integer is INTEGER
```

```
Lex Character is CHARACTER
Lex Float is FLOAT
Lex Void is VOID
Lex { is OPEN_BLOCK
Lex $bool is PERL_IDEN_SCALAR
Lex && is AND
Lex $bool2 is PERL_IDEN_SCALAR
Lex || is OR
Lex $bool3 is PERL_IDEN_SCALAR
Lex } is CLOSE_BLOCK
Lex ( is OPEN_FUNCTION
Lex ) is CLOSE_FUNCTION
Lex @array1 is PERL_IDEN_ARRAY
Lex [ is OPEN_ARRAY
Lex ] is CLOSE_ARRAY

Process finished with exit code 0
```

Question 2:

main.c:

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

const int ARRAY_SIZE = 1000;

void subscripting() {
  int arr[ARRAY_SIZE][ARRAY_SIZE];
  int x;
  for (int i = 0; i < ARRAY_SIZE; i++) {
    x = arr[i][i];
  }
}

void pointer() {
  int arr[ARRAY_SIZE][ARRAY_SIZE];
  int x;
  for (int i = 0; i < ARRAY_SIZE; i++) {
    int *p = *(arr + i);
    x = *(p + i);
  }
}

int main() {
  struct timespec tStart, tEnd;
  // Subscripting
  clock_gettime(CLOCK_REALTIME, &tStart);
  subscripting();
  clock_gettime(CLOCK_REALTIME, &tEnd);
  printf("Subscripting: %ld nano seconds\n", tEnd.tv_nsec -
tStart.tv_nsec);
  // Pointer
  clock_gettime(CLOCK_REALTIME, &tStart);
  pointer();
  clock_gettime(CLOCK_REALTIME, &tEnd);
  printf("Pointer: %ld nano seconds\n", tEnd.tv_nsec - tStart.tv_nsec);

}
```

Output:
Subscripting: 1353428 nano seconds
Pointer: 7877 nano seconds

Answer:
As we can see, using pointers to access an array is way more efficient. This is because the pointer has direct address to access the memory. For the reliability, I think using array is more reliable. Because if we use pointer, we can accidently access the part of memory that is out of the array's boundary.

Question 3:

main.pl:

```
# Name: Minh Nguyen
# Panther Id: 002-46-4288

my %hash;
my $ITERATION = 1500000;

sub generateName {
   my $str = "";
   for (my $i = 0; $i < 3; $i = $i + 1) {
      my $x = ord('A') + (int rand(25));
      my $c = chr($x);
      $str = "$str$c";
   }
   return $str;
}

sub generateAge() {
   return (1 + int rand(99));
}

sub usingHash() {
   for (my $i = 0; $i < $ITERATION; $i = $i + 1) {
      my $name = generateName();
      my $age = generateAge();
      $hash{$name} = $age;
   }
}

sub notUsingHash () {
   for (my $i = 0; $i < $ITERATION; $i = $i + 1) {
```

```
        my $name = generateName();
        my $age = generateAge();
    }
}

# Using hash
my $start = time;
usingHash();
my $duration = time - $start;
print "Using hash: $duration seconds\n";

# Not using hash
$start = time;
notUsingHash();
$duration = time - $start;
print "Not using hash: $duration seconds\n";
```

Output:

```
$perl main.pl
Using hash: 5 seconds
Not using hash: 4 seconds
```

Answer:

As we can see, the program which runs without using hash has worse performance. Using hash improves in efficiency and readability of the code.

Question 4:

Answer:

Naming and Structuring are the features of the compilation process that allow us to determine the reference environment for any random line of code in the program. Linker is one of the steps in this process. When we create a scope, a name is also created in a tree-like structure. Every declared variable is inside this scope will be assigned to this scope.

Scoping affects the reference environment. Static scoping the environment would change the static referencing. This is because in static scoping, only the variables can be referred that are in static reference environment. But it cannot refer to the variables of dynamic ancestors. If the program is scoped dynamically, the environment will be local.

Question 5:

Answer:

When I was designing my own lexical analyzer (question 1), I need to specify some reserved words for type declarations. These words need to be exactly the same (case sensitive), and they need to stand alone by itself. If the users choose to use a reserve word as an identifier, they

need to specify it as an identifier (use $, %, or @) to let the analyzer know that it can ignore the case where it matches the typing with the reserved words.

Question 6:

EBNF:
If-else statement:
<IF_STMT> → if "(" <LOGIC_EXP> ["&&" <LOGIC_EXP>]* ["||" <LOGIC_EXP>]* ")" "{"
<BODY> "}"
[else "{" <BODY> "}"]*
<BODY → [<STMT>]+

While statement:
<WHILE_STMT> → while "(" <LOGIC_EXP> ["&&" <LOGIC_EXP>]* ["||" <LOGIC_EXP>]* ")" "{"
<BODY> "}"
<BODY → [<STMT>]+

Logical/Mathematical Expression:
<EXP> → <TERM> [+ | -] <TERM>
<TERM> → < FACTOR> [* | /] < FACTOR>
<FACTOR> → JAVA_INT_LIT | ID \ "(" <EXP> ")"

Mathematical Assignment Statement:
<ASSIGN_STMT> → ID "=" <EXP>
<EXP> → <TERM> [+ | -] <TERM>
<TERM> → < FACTOR> [* | /] < FACTOR>
<FACTOR> → JAVA_INT_LIT | ID \ "(" <EXP> ")"

Question 7:

Answer:
RDA has the limitation that it needs to strictly follow LL grammars. Also, it must pass the pairwise-disjointness test, and it cannot have left-hand recursion, which means it cannot refer to itself. A invalid example would be <STMT> → <STMT>. This can cause an infinite loop.
In order to allow either a while statement, if statement or assignment statement, <STMT> has to be either <WHILE_STMT>, <IF_STMT>, or <ASSIGN_STMT>. Or:
<STMT> → (<WHILE_STMT> | <IF_STMT> | <ASSIGN_STMT) +

Question 8:

Question8.py:

```
# Name: Minh Nguyen
# PantherID: 002-46-4288

def func1 ():
    a = 1
    b = 1
    c = 1
    print("Inside func1:")
    print("a = " + str(a))
    print("b = " + str(b))
    print("c = " + str(c))

    def func2 ():
        b = 2
        print("Inside func2:")
        print("a = " + str(a))
        print("b = " + str(b))
        print("c = " + str(c))

        def func3 ():
            c = 3
            print("Inside func3:")
            print("a = " + str(a))
            print("b = " + str(b))
            print("c = " + str(c))
            # End func3()

        func3() # Call func3() inside func2()

        # End func2()

    func2() # Call func2() inside func1()

# End func1()

func1() # Main calling func1()
```

Output:

Inside func1:
a = 1
b = 1
c = 1
Inside func2:
a = 1
b = 2
c = 1

Inside func3:
a = 1
b = 2
c = 3

Process finished with exit code 0