Homework 3
Akash Dansinghani
Professor Umoja
Due: 11/06/20
Programming Language Concepts

1.
```
void main() {
        int a, b, c;
        fun1();
}
```

The main calls function a, b calls fun1, c calls fun2

```
void fun1(void) {
        int b,c,d;
        fun2();
}
void fun2(void) {
        int c,d,e;
        fun3();
}
void fun3(void) {
        int d,e,f;
        # calls a,b,c,d,e,f where a,b,c are in main(), b,c,d are in fun1(), c,d,e are in fun2(), d,e,f are in fun3()
}
```

a.

a – main(), b calls fun1(), c calls fun2(), d, e, f calls fun3()

b.

a – main(), b,c calls fun1(), d, e, f calls fun3()

c.

a – main(), e, f calls fun3(), b,c,d calls fun1()

d.

a – main(), e, f calls fun3(), b,c,d calls fun1()

e.

a – main(), b calls fun1(), f calls fun3(), c,d,e calls fun2(),

f.

a – main(), f calls fun3(), e calls fun2(), b,c,d calls fun1(),

2.

A = 15;

X = 1;

Y = 3;

Z = 5;

These variables are global variables that are defined and can be called from subroutine programs. Local variables are defined in the program and can only be accessed if called in that specific subroutine program.

Def sub1():

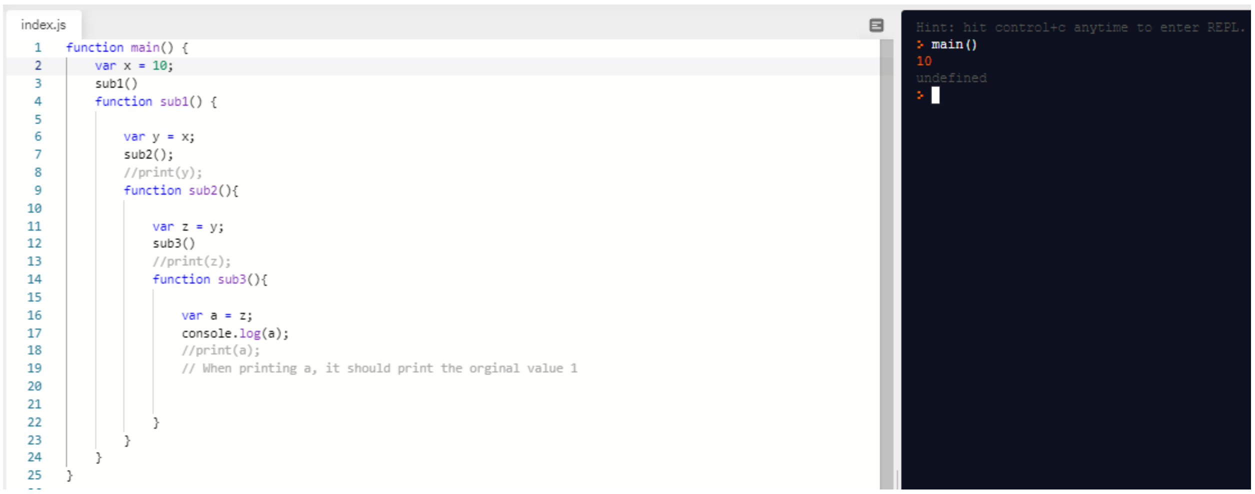
```
    A = 7;

    Y = 9;

    Z = 11;
```

Program sub1() has access to A = 7; , X = 1; , Y = 9; , Z = 11;. A, Y, and Z are all defined in the program but X is not. However, it is a global variable and is defined in the main, so it is able to be called from sub1(). The values for A, Y, and Z are different than the global variables which is fine because they have specific variables that are defined in sub1() , but since X is not defined in the program; it can use the global variable. <u>We can not manipulate the variable x but we can access and print the value of x because it is a read-only global variable.</u>

Program sub2() has access to X = 1; , (called global variable using scoping), A = 13; , X = 15; , W = 17;. A, X, and W are all defined in the program. However, x is a global variable and is defined in the main, so it is able to be called from sub2() and be manipulated. The values for A, X, and W are different than the global variables which is fine because they have specific variables that are defined in sub2() , but the variable x in the main is able to be manipulated (changed or altered) using the keyword "global" which gives access to the global variable (static scoping). So x = 15; would not be changed, just x = 1; from the global variable.

Program sub3() has access to nonlocal a; , b = 21; , Z = 23;. A, B and Z are all defined in the program. However, the first line in the subroutine program 3, a grants the program access to nonlocal a and since it is nested in the subroutine of sub2() , it is able to manipulate a = 13; from that program. The values for A, B, and Z are different than the global variables which is fine because they have specific variables that are defined in sub2() , but the variable a in the sub2() is able to be manipulated (changed or altered) using the keyword “nonlocal” which gives access to sub2(). The nonlocal variable a = 13; is able to be changed but a = 19; that is a local variable would not be changed.

3.



index.js

```
function main() {
    var x = 10;
    sub1()
    function sub1() {

        var y = x;
        sub2();
        //print(y);
        function sub2(){

            var z = y;
            sub3()
            //print(z);
            function sub3(){

                var a = z;
                console.log(a);
                //print(a);
                // When printing a, it should print the orginal value 1


            }
        }
    }
}
```

```
Hint: hit control+c anytime to enter REPL.
> main()
10
undefined
>
```

4. HW3Q4.py file

5.

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

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

HW3Q5.JAVA file (for code)

6. Java:

```
  // int temp = 0;
   //int val = 3;
   //int sum;
   //sum = temp + val;
   //System.out.println("Adding to values = " +sum);
   int sum, temp, val
   sum = temp + val
```

Identifiers: sum, temp, val

Operators: =, +

The type of binding this has is “Compiler Time Binding”. The address is loading time binding if using static scoping. If not, it is considered runtime binding. The value uses runtime binding. The scope uses static scoping since we are using compiler time binding. The life time for the program can vary on a multitude of things. Since we are using local variables that are defined in the class, it is based on the method call. The meaning of the equal or “=” operator is language design time binding and the meaning of the addition or “+” operator is compiler design time binding to find the sum.

7. Dynamic type binding is closely related to implicit heap – dynamic variables because they both relate to the assignment and statement with implicit heap – dynamic variables types are able to be generated when values are received to the variables which will happen once a program compiles and is able to have a runtime. Given that they are given their type at runtime, the variables are classified as a dynamic type binding.

8. Data manipulation is great for subprograms where an operation is executed on a variable and the function can be called multiple times if different variables need to be run through the subprogram it can be manipulated. The function only has to return it from the subprogram instead of initializing it as a parameter like in most programs (C# is a great example).