2 hours (maximum)
Closed Book

- You may use both sides of one full sheet (8.5x11) of paper covered with any notes you like.
- This exam has 14 pages, including this cover page and a scratch page at the end. Do all your work on these exam sheets; use the backs of the pages if needed.
- Be specific and clear in your answers.
- Show all your work if you wish to be considered for partial credit.
- If you use the back pages for scratch space, cross out your scratch work before you submit the exam.

<table>
<thead>
<tr>
<th>Question</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

Name: 

Email: 

DO NOT TURN TO NEXT PAGE UNTIL YOU GET PERMISSION
1. [10 points] Nested scopes problem.
   (a) What is the output for the below C program?

```c
int main() {
    int x, y, z;
    x = 1;
    {
        y = 2;
        int z = 3;
        printf("%d, %d, %d\n", x, y, z);
    }
    z = 3;
    x = 4;
    {
        int z = 5;
        printf("%d, %d, %d\n", x, y, z);
        z = 6;
        y = 7;
    }
    {
        int y = 8;
        {
            int y = 9;
            printf("%d, %d, %d\n", x, y, z);
        }
        printf("%d, %d, %d\n", x, y, z);
    }
}
```

(b) Draw the activation record for main. Show where all of the variables are located and what their values are at the end of the program. You do not need to show where the frame pointer and stack pointer are pointing. You also do not need to show where the old frame pointer and activation records are stored.
2. [10 points] Translating to 3-address code.
For the following two C code snippets, write the 3-address code that implements the snippet. You can assume that each variable, including the “this” variable, can be accessed directly in 3-address code (e.g. \( a = a + 1 \)). Also, you can create as many temporaries as you need (e.g. \( t_1 = t_2 + 1 \)).

\[
do \{ a = a + 1; \} \text{ until (a>10);}\]

\[
s \&\& q \quad \text{\( \backslash\backslash \) put the result in \( t_1 \)}
\]
3. [10 points] Reading Assembly. Below is assembly code generated by “gcc -S” on a 64-bit machine for the given C program. (a) Indicate separately which assembly instructions are for the body of the while loop and which instructions are for the while loop test and loop. (b) For each instruction in the while loop test and loop, write a brief summary (i.e. goto .L1 or x = y + 1) next to the instruction.

```c
#include <stdio.h>
int main() {
    int i;
    int b = 10;
    while ( b > 2 ) {
        b = b - 1;
        for (i=4; i<7; i++) {
            printf("%d\n", i*b);
        }
    }
}
```

```assembly
.LC0:
    .string "%d\n"
.text
.globl main
    .type main, @function
main:
.LFB2:
pushq %rbp
.LCFI0:
    movq %rsp, %rbp
.LCFI1:
    subq $32, %rsp
    jmp .L2
.L5:
    subl $1, -4(%rbp)
    movl $4, -8(%rbp)
    jmp .L3
.L4:
    movl -8(%rbp), %eax
    movl %eax, %esi
    imull -4(%rbp), %esi
    movl $.LC0, %edi
    movl $0, %eax
    call printf
    addl $1, -8(%rbp)
.L3:
    cmpl $6, -8(%rbp)
    jle .L4
.L2:
    cmpl $2, -4(%rbp)
    jg .L5
    leave
    ret
```
4. [20 points] Memory Layout. For each of the two MiniJava programs listed below, show what values are in
the stack and the heap at runtime at the point indicated for each program. Show where the frame pointer
will be pointing, show where the old %ra and %fp registers are stored, and show where all of the parameters,
locals, and member variables are stored and their current value. For the values of pointers (including $fp
and $sp) draw arrows to indicate what address the pointer variable contains.

(a) For this program show the memory layout before the third call to recurse returns. Show the value that
the third call to recurse will be putting into the return value slot.

class RecursiveUpDown {
    public static void main(String[] a){
      System.out.println(new Foo().Recurse(8, 5));
    }
}

class Foo {
    boolean why;

    public int Recurse(int down, int up){
      int diff;

      diff = down-up;
      System.out.println(diff);
      if (diff < 0)
        diff = 42 ;
      else
        diff = this.Recurse(down-1, up+1) ;
      return diff ;
    }
}
(b) For this program show the memory layout before the call to foo returns. Fill in foo’s return value slot.

class InheritFinal {
    public static void main(String[] a) {
        System.out.println((new Sub().baz())[1]);
    }
}

class Super {
    int a;
    public int[] baz() {
        int[] y;
        y = new int[2];
        y[0] = 7;
        y[1] = this.foo(4, 5);
        return y;
    }
    public int foo(int p, int q) {
        a = p * q;
        return a;
    }
}

class Sub extends Super {
    int a;
    public int[] baz() {
        int[] y;
        a = 2;
        y = new int[3];
        y[2] = 42;
        y[1] = this.foo(2, 7);
        return y;
    }
}
5. [10 points] Implementing a new language feature. 
Describe how you would extend the MiniJava compiler so that it includes named parameters. Briefly describe how each stage of compilation would be affected. Assume that the actual parameters must still be evaluated from left to right.

The following is an example of named parameter usage:

```java
class NamedParams {
    public static void main(String[] whatever) {
        System.out.println(new Foo().testing());
    }
}

class Foo {
    int a;
    public int testing() {
        System.out.println(this.baz(f=new Foo(), a = 42, b = true));
    }
    public int baz(int a, boolean b, Foo f) {
        ...
    }
}
```
6. [20 points] LR Parsing Table.

a) For the following grammar:

(0) \( S \rightarrow L \)
(1) \( S \rightarrow \epsilon \)
(2) \( L \rightarrow a \land L \)
(3) \( L \rightarrow a \)

Draw the LR(1) states and transitions between those states. Then fill in the empty LR parse table on the next page. The parse table contains the number of states you will actually need.
b) Is this grammar LR(0)? Why or why not?
7. [10 points] Semantic Analysis in MiniJava.

(a) Label each expression node in the AST on the following page with its type (int, int [], boolean, or class(classname)). Make sure to label each expression node in terms of what its type should be even if its operands have type errors.

(b) On the next page, mark each AST node in which the check types visitor should detect an error or errors. Next to the relevant AST nodes, write one phrase that describes each error.

(c) On this page, write pseudo-code that more generally implements type checking for the ArrayAssignStatement node. Make sure to indicate whether you are overriding the in, out, or case method for each ast node. Assume that you can access the kind of type information derived in part (a) of this question by using a HashMap variable called nodeType. For example, nodeType(index) or nodeType(node.getExp()) would return the type of the index expression when visiting the ArrayAssignStatement node.
Program

MainClass

TopClassDecl

Token

FinalSemantic

Token

v PrintStatement

CallExp

NewExp

Token
go

IntegerExp

Token
Tst

Token
22

Token
42

MethodDecl

ArrayAssignStatement

Token

y

Formal

Token

y

Token
go

BoolType

Token
gO

ArrayType

Token

y

Token

y

Token

y

Token

true

CallExp

Token

go

IdExp

Token

y

Token

y

Token

y

<table>
<thead>
<tr>
<th>Productions</th>
<th>Semantic Rules</th>
</tr>
</thead>
</table>
| $S \rightarrow(assign$ | $S.code = assign.code$
| $S \rightarrow (B) S_1$ | $\text{if } (S = \text{root}) \text{ then } S.next = \text{newlable()}$
| $B \rightarrow E_1 \text{ rel } E_2$ | $\text{test} = E_1.addr \text{ rel } op E_2.addr$
| $B \rightarrow B_1 \parallel B_2$ | $B_1.true = \text{if } B.true \neq \text{fall} \text{ then } B.true \text{ else newlabel()}$

Using the above grammar and its associated semantic rules to do the following:

(a) Making sure to leave room for later annotations, draw an abstract syntax tree for the following code segment:

```latex
\begin{verbatim}
if ( a > b || c < d || e == f ) {
    v = 42;
}
\end{verbatim}
```
(b) Annotate the AST on the previous page with the inherited attributes specified by the semantic rules.

(c) Write the 3-address code for the statement by evaluating the synthesized attribute in the semantic rules.