Login Name _____ **Name** _____

Student ID _____ **Signature** _____

By filling in the above and signing my name, I confirm I will complete this exam with the utmost integrity and in accordance with the Policy on Integrity of Scholarship.

Final
CSE 131
Winter 2013

Page 1	_____	**(29 points)**
Page 2	_____	**(30 points)**
Page 3	_____	**(24 points)**
Page 4	_____	**(25 points)**
Page 5	_____	**(21 points)**
Page 6	_____	**(38 points)**
Page 7	_____	**(27 points)**
Page 8	_____	**(23 points)**
Page 9	_____	**(16 points)**
Page 10	_____	**(18 points)**
Page 11	_____	**(21 points)**
Subtotal	_____	**(272 points) = 100%**
Page 12 Extra Credit	_____	**(14 points) [5+% Extra Credit]**
Total	_____	

This exam is to be taken by yourself with closed books, closed notes, no electronic devices.
You are allowed both sides of an 8.5"x11" sheet of paper handwritten by you.

1. Given the following CUP grammar snippet (assuming all other Lexing and terminals are correct):

```
Expr ::=   Des {: System.out.println("A"); :} AssignOp Expr {: System.out.println("K"); :}
     |     Des {: System.out.println("B"); :}
     ;

Des  ::=   T_STAR {: System.out.println("C"); :} Des {: System.out.println("L"); :}
     |     T_PLUSPLUS {: System.out.println("D"); :} Des {: System.out.println("M"); :}
     |     T_AMPERSAND {: System.out.println("E"); :} Des {: System.out.println("N"); :}
     |     Des2 {: System.out.println("F"); :}
     ;

Des2 ::=   Des2 {: System.out.println("G"); :} T_PLUSPLUS {: System.out.println("O"); :}
     |     Des3 {: System.out.println("H"); :}
     ;

Des3 ::=   T_ID {: System.out.println("I"); :}
     ;

AssignOp ::=  T_ASSIGN {: System.out.println("J"); :}
     ;
```

What is the output when parsing the follow expression (you should have 26 lines/letters in your output):

```
*++x = *y++ = &z
```

Output

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

In the above grammar, what is the associativity of the operators in the second production rule (the Des ::= rule)? _____

If variable `z` is defined to be type `int`, what types must variables `x` and `y` be defined to be for this statement to be semantically correct?

_____ x; _____ y;

2. Given the following Reduced-C code fragment:

```
function : int foo( int & x, int * y, int z ) { /* Body of code not important for this question */ }

function : int main()
{
  int a;
  int b = 42042;
  int c = b;

  a = foo( c, &a, b );

  return c;
}
```

Complete the SPARC Assembly language statements that might be emitted by a compliant Reduced-C compiler from this quarter for function main(). Allocate, store, and access all local variables on the Stack. See comments.

```
        .section __________

        .global __________
        .align 4

_______:
        set     ___________________, %g1

        save    ___________________, %g1, ___________________

        /* Initialize the local variables that have explicit initialization in this stack frame */

        set     _____________, %o0

        st      %o0, _____________                  ! int b = 42042;

        ld      _____________, %o0

        st      %o0, _____________                  ! int c = b;

        /* Set up the 3 actual arguments to foo() */

        _______  ___________________________, %o0  ! large blank can be one or two operands

        _______  ___________________________, %o1

        _______  ___________________________, %o2

        call    foo                                 ! Call function foo()

        __________

        st       _____________, [%fp - 16]          ! Save return value into local temp1

        /* Copy saved return value stored in temp1 into local var a */

        _______  [%fp - 16], ___________________

        _______  _____________, _____________       ! a = foo( ... );

        /* return c; */

        ld       _____________, _____________       ! return c;

        ___________

        ___________

        MAIN_SAVE = -(92 + _______) ________ ________  ! Save space for 3 local vars + 1 temp
```

3. In object-oriented languages like Java, determining which method code/instructions to bind to (to execute) is done at run time rather than at compile time (this is known as dynamic dispatch or dynamic binding). However, the name-mangled symbol denoting a particular method name is determined at compile time. Circle the page number in the lower right of this page for an extra point. Given the following Java class definitions, specify the output of each print() method invocation - **Use the letters A-F below to denote which string is printed**.

```
class C
{
    public void print(C p)
    {
        System.out.println("C 1");
    }
}
```

```
class CPP extends C
{
    public void print(C p)
    {
        System.out.println("CPP 1");
    }

    public void print(CPP p)
    {
        System.out.println("CPP 2");
    }
}
```

```
class C_SHARP extends CPP
{
    public void print(C p)
    {
        System.out.println("C_SHARP 1");
    }

    public void print(CPP p)
    {
        System.out.println("C_SHARP 2");
    }

    public void print(C_SHARP p)
    {
        System.out.println("C_SHARP 3");
    }
}
```

```
A) C 1
B) CPP 1
C) CPP 2
D) C_SHARP 1
E) C_SHARP 2
F) C_SHARP 3
```

```
public class Overloading_Final_Exam {
    public static void main (String [] args) {

        C lang1 = new C();
        C lang2 = new CPP();
        C lang3 = new C_SHARP();
        CPP lang4 = new CPP();
        CPP lang5 = new C_SHARP();
        C_SHARP lang6 = new C_SHARP();


        lang1.print( lang1 );                          _____________________

        lang2.print( lang2 );                          _____________________

        lang3.print( lang3 );                          _____________________

        lang4.print( lang4 );                          _____________________

        lang5.print( lang5 );                          _____________________

        lang6.print( lang6 );                          _____________________


        lang4.print( (C) lang6 );                      _____________________

        lang5.print( (CPP) lang6 );                    _____________________

        lang6.print( (C_SHARP) lang6 );                _____________________


        lang1.print( lang6 );                          _____________________

        ( (CPP) lang3).print( (CPP) lang6 );           _____________________

        ( (C_SHARP) lang3).print( (C) lang6 );         _____________________

    }
}
```

Now <u>remove</u> the entire `print(C p) {}` method in `class CPP` and <u>remove</u> the entire `print(CPP p) {}` method in `class C_SHARP`. Specify the output of each print() method <u>with these changes</u> below.

```
lang1.print( lang1 );                          _____________________

lang2.print( lang2 );                          _____________________

lang3.print( lang3 );                          _____________________

lang4.print( lang4 );                          _____________________

lang5.print( lang5 );                          _____________________

lang6.print( lang6 );                          _____________________


lang4.print( (C) lang6 );                      _____________________

lang5.print( (CPP) lang6 );                    _____________________

lang6.print( (C_SHARP) lang6 );                _____________________


lang1.print( lang6 );                          _____________________

( (CPP) lang3).print( (CPP) lang6 );           _____________________

( (C_SHARP) lang3).print( (C) lang6 );         _____________________
```

4. Given the following Reduced-C code below, fill in the blanks of the compile error that should be reported according to this quarter's Project I spec. Use the letters associated with the words in the box below.

```
typedef float F1;
typedef F1 F2;
typedef int I1;
typedef I1 I2;

I1 x;
I2 y;
F2 z;

x = z = y; // Compile error reported here. Assume this stmt is inside a function.
```

A. I1	B. I2
C. int	D. F1
E. F2	F. float
G. 5-hour Energy	H. Sleep
I. equivalent	J. modifiable
K. addressable	L. assignable

Value of type ____ not ____ to variable of type ____ .

Given the following test.rc fragment:

```
bool a;
int b;

b = a++ + ++a;
```

A) WNBT
B) Prints one error - for the leftmost a++
C) Prints one error - for the rightmost ++a
D) Prints two errors - for both a++ and ++a

Which is true based on this quarter's Project spec? _____

Consider the following struct definitions in Reduced-C (similar to C/C++). Specify the size of each struct on a typical RISC architecture (like ieng9) or 0 if it is an illegal definition.

```
structdef FOO2 {
  FOO2 *  a;
  float   b;
  function : void bar( FOO2 &x)
  {
    int[10] y;
  }
  int[2]  c;
  int     d;
};
```

Size _____

```
structdef FOO3 {
  int  * a;
  float  b;
  function : void bar()
  {
    int x = *this.d;
  }
  FOO3 * c;
  int  * d;
};
```

Size _____

```
structdef FOO1 {
  int    a;
  float  b;
  function : void bar()
  {
    FOO1 *x;
  }
  int    c;
  int[4] d;
};
```

Size _____

Using the Reduced-C Spec (which closely follows the real C language standard), given the definitions below, indicate whether each expression evaluates to either a

1) R-val 2) Non-Modifiable L-val 3) Modifiable L-val

```
function : int * foo1() { /* Function body not important. */ }
function : int & foo2() { /* Function body not important. */ }
float[9] a;
float x;
const float y = 5.5;
float *p = &x;
```

____ foo2() ____ 420 ____ *foo1() ____ ++foo2() ____ y ____ (int *)p

____ a ____ a[2] ____ foo2 ____ *(int *)&x ____ &*p ____ foo1()

____ *p++ ____ ++*p ____ *++p ____ --*p++ ____ a[2]++ ____ (*p)++

5. What gets printed in the following C++ program (just like Reduced-C without "function : " in front of each function definition)? If a value is unknown/undefined or otherwise cannot be determined by the code given, put a question mark ('?') for that output. Hint: Draw stack frames!

```
int a = 4;
int b = 1;
int c;
int mo;

int & fubar( int x, int & y, int * z )
{
   static int m = x;
   x = x + 2;
   y = y + 2;
   *z = *z + 2;
   mo = ++m;

   return x;
}

void foo1( int & d, int * e, int f )
{
   d = d + 3;
   *e = *e + 3;
   f = f + 3;

   cout << a << endl;    ________

   cout << b << endl;    ________

   cout << c << endl;    ________

   cout << d << endl;    ________

   cout << *e << endl;   ________

   cout << f << endl;    ________

   cout << mo << endl;   ________

   cout << fubar( d, d, &d ) << endl;   ________

   cout << fubar( *e, *e, e ) << endl;  ________

   cout << fubar( f, f, &f ) << endl;   ________

   cout << a << endl;    ________

   cout << b << endl;    ________

   cout << c << endl;    ________

   cout << d << endl;    ________

   cout << *e << endl;   ________

   cout << f << endl;    ________

   cout << mo << endl;   ________
}

int main()
{
   foo1( a, &b, c );

   cout << a << endl;    ________

   cout << b << endl;    ________

   cout << c << endl;    ________

   cout << mo << endl;   ________

   return 0;
}
```

6. Using the load/load/compute/store and internal static variable paradigms recommended in class and discussion sections, complete the SPARC Assembly language statements that might be emitted by a compliant Reduced-C compiler from this quarter for function foo(). Store all formal params on the Stack.

```
function : int foo( int *x, int y, int & z )
{
  static int c = z;

  *x = c - y;

  return z;
}
```

```
        .section        "____________"
        .global         ____________
        .align          4
foo:
        set     foo.SAVE, %g1
        save    %sp, %g1, %sp

        st      %i0, _______________

        st      %i1, _______________

        st      _________, [%fp + 76]

        .section        "____________"
        .align          4
.foo_c:
        .skip           4
.foo_c_flag:
        .skip           4
        .section        "____________"
! Check if internal static var c has
! already been initialized

        set     ____________________, %o0
        ld      [%o0], %o0

        cmp     __________, __________

        ______  .L1             ! skip init
        nop
! set flag to skip all further inits
        set     ______, %o0

        set     .foo_c_flag, %o1

        st      _______, ___________
! Init internal static var c for 1st time

        ld      _______________, %o0

        ______  _______________, %o0

        st      %o0, [%fp - 4] ! tmp1 = z
        ld      [%fp - 4], %o0

        set     ____________________, %o1
! c = z
        ______  _________, __________
```

```
.L1:

! Perform *x = c - y; block

! c - y
        set     _______________, %o0

        ______  [%o0], %o0          ! c

        ld      _______________, %o1 ! y

        ______  %o0, %o1, %o0       ! c - y

! tmp2 <- (c - y)
        st      __________, [%fp - 8]

! previous result from tmp2
        ld      [%fp - 8], %o0

! get param x
        ld      _______________, %o1

! *x = c - y; (store tmp2 into *x)

        _______ %o0, _____________


! return z;
        ld      _____________, %o0

        ld      _____________, %o0

        ______  %o0, __________

        _________

        _________

! save space for 2 temporaries on stack

foo.SAVE = -(92 + _____) _____ ______
```

7. Given the C array declaration

```
int a[4][2];
```

Mark with an **A** the memory location(s) where we would find

```
a[2][1]
```

a:

low memory high memory

Each box represents a byte in memory.

If `a[0][0]` is allocated at memory location 8000_{10}, what value does `&a[3][1]` evaluate to? _____

Using the Right-Left rule write the C definition of a variable named foo that is a pointer to an array of 4 elements where each element is a pointer to a function that takes a pointer to a struct Fubar as the single parameter and returns a pointer to a 7x3 2-D array where each element is a pointer to a pointer to a struct RT.

Identify whether each of the following will cause an underlying bit pattern change.

```
int a = 5;
float b = -4.20;
int * ptr1;
float * ptr2;
void foo( float x, float & y ) { /* … */ }
```

1) Yes – Underlying bit pattern change
2) No – No underlying bit pattern change

`ptr1 = (int *) & b;`	_____	`ptr2 = (float *) ptr1;`	_____
`a = (int) b;`	_____	`foo( a, b );`	_____
`b = a;`	_____	`a = * (int *) & b;`	_____

According to this quarter's Reduced-C grammar, what two constructs must be uppercase symbols?

_____ _____

Given the variable definitions: `int * x, y;`

What type is x and what type is y in Reduced-C? _____ x _____ y

What type is x and what type is y in C/C++? _____ x _____ y

Using Reduced-C syntax, define a pointer to an array of 8 bools named `fool` such that

```
bool x = true;

(*fool)[7] = x;
x = (*fool)[7];
```

are valid expressions. This will take two lines of Reduced-C code.

8. Given the following program, specify the order of the output lines when run and sorted by the address printed with the %p format specifier on a Sun SPARC Unix and Linux system. For example, which line will print the lowest memory address, then the next higher memory address, etc. up to the highest memory address?

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

void foo1( int *, int ); /* Function Prototype */
void foo2( int, int * ); /* Function Prototype */

struct foo {int a; int b;} a = { 1, 2 };

int main( int argc, char *argv[] )
{
   int b;
   int c;

   foo2( b, &c );

/*  1 */ (void) printf( "1: b --> %p\n", &b );
/*  2 */ (void) printf( "2: argc --> %p\n", &argc );
/*  3 */ (void) printf( "3: c --> %p\n", &c );
/*  4 */ (void) printf( "4: argv --> %p\n", &argv );
}

void foo1( int *d, int e )
{
   int f = 411;
   int g;

/*  5 */ (void) printf( "5: f --> %p\n", &f );
/*  6 */ (void) printf( "6: a.b --> %p\n", &a.b );
/*  7 */ (void) printf( "7: malloc --> %p\n", malloc(50) );
/*  8 */ (void) printf( "8: e --> %p\n", &e );
/*  9 */ (void) printf( "9: foo2 --> %p\n", foo2 );
/* 10 */ (void) printf( "10: d --> %p\n", &d );
/* 11 */ (void) printf( "11: a.a --> %p\n", &a.a );
/* 12 */ (void) printf( "12: g --> %p\n", &g );
}

void foo2( int h, int *i )
{
   int j[3];
   static int k;

   foo1( i, k );

/* 13 */ (void) printf( "13: j[1] --> %p\n", &j[1] );
/* 14 */ (void) printf( "14: h --> %p\n", &h );
/* 15 */ (void) printf( "15: k --> %p\n", &k );
/* 16 */ (void) printf( "16: i --> %p\n", &i );
/* 17 */ (void) printf( "17: j[0] --> %p\n", &j[0] );
}
```

_____ smallest value (lowest memory address)

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____ largest value (highest memory addresses)

You are compiling foo1.c and foo2.c together with gcc. If foo1.c has the following statement

```
extern int a;
```

indicate whether each of the following would cause a linkage editor error or not if put separately (one at a time) in foo2.c?

A) Yes - Linkage Editor Error
B) No - No Linkage Editor Error

____ `int a = 42;`

____ `extern int a;`

____ `static double a;`

____ `extern void a( char * );`

____ `int a( float b ) { return (int)b; }`

____ `static int a( float b) { return (int)b; }`

9. Given the following C++ definitions (similar to Reduced-C)

```
struct S1 { int a; };
struct S2 { int a; };

void foo1( int a ) { }
void foo2( S1 &a ) { }

typedef int T1;
typedef T1 T2;
typedef S1 T3;
typedef T3 T4;

S1 a;
S2 b;
T1 c;
T2 d;
T3 e;
T4 f;
float g;
```

indicate whether each of the following statements will cause a compiler error or not.

A) No Error
B) Error

```
  c = g; _____          foo2( e ); _____
  a = e; _____          foo1( d ); _____
  a = f; _____          foo1( g ); _____
  a = b; _____          foo2( a ); _____
  c = d; _____          foo2( f ); _____
                        foo2( b ); _____
```

Given the following C type definitions

```
struct foo {            struct fubar {
  short  a;               char       e[5];
  char   b;               int        f;
  double c;               struct foo g;
  short  d;               int        h;
};                      };

struct fubar fubaz;
```

What is the `offsetof( struct fubar, g.c )`? _____ What is the `sizeof( struct fubar )`? _____

If `struct fubar` had been defined as `union fubar` instead, what would be the `sizeof(union fubar)`? _____

What is the resulting type of the following expression?

```
* (short *) & ( ( (struct foo *) & fubaz.e ) -> d ) _________________
```

Write the equivalent expression that directly accesses this value/memory location without all the fancy casting and `&` operators.

```
fubaz._________________
```

10. Given the following Reduced-C definitions:

```
function : float & foo1( float & a ) { return a; }
function : float & foo2( float a )   { return a; }
function : float   foo3( float & a ) { return a; }
function : float   foo4( float a )   { return a; }

float x;  /* global variables */
int y;
```

For each of the following statements, indicate the type of error (if any) that should be reported according to the Project I spec for this quarter. Use the letters associated with the available errors in the box below.

```
x = foo1( 4.2 );                ____
y = foo3( x );                  ____
x = foo4( foo4( x ) );          ____
x = foo1( x + y );              ____
x = foo2( &x );                 ____
x = foo3( y );                  ____
foo4( x ) = foo1( x );          ____
foo1( x ) = (int) foo4( x );    ____
y = (int)foo3( (float)y );      ____
*(int *)&foo2( 42 ) = y;        ____
&foo2( x ) = (float *)&y;       ____
```

A) No Error
B) Argument passed to reference param is not a modifiable L-val
C) Argument not assignable to value param
D) Argument not equivalent to reference param
E) Left-hand operand is not assignable (not a modifiable L-val)
F) Right-hand-side type not assignable to left-hand-side type

Give two examples where you used initialization guards in your compiler.

1)

2)

Why did you need these initialization guards?

Why did you need to name mangle static variable names in your compiler?

Function pointers are two four-byte entities. Explain what each of the four-bytes is used for.

1)

2)

We recommended you store all the parameters into the parameter section of the stack frame on the stack in the prologue of each function instead of keeping them in their respective %i registers. Why?

11. Match the compilation process with the various tasks done in the compilation sequence.

1) C++ Preprocessor 2) C++ Compiler 3) Assembler 4) Linkage Editor 5) Loader

_____ performs syntax analysis on its input high-level language (HLL).

_____ resolves undefined external symbols with defined global symbols in other modules.

_____ translates high-level language (HLL) code into assembly code.

_____ zero fills the BSS segment in memory.

_____ puts globally defined symbols in the export list of the resulting object file.

_____ performs semantic analysis on its input high-level language (HLL).

_____ takes an executable file on disk and makes it ready to execute in memory.

_____ translates assembly code into machine code.

_____ combines all object modules into a single executable file.

_____ performs name mangling of function names.

_____ expands # directives from its input high-level language (HLL).

Use virtual register notation for each of the following.

Change the following instruction into three instructions which are most likely a time improvement over the single instruction when it comes to actual execution time.

```
r2 = r4 * 126
```

_____

_____

_____

What term describes this particular kind of peephole optimization?

Change the following instructions into two instructions which are most likely a time improvement over the set of instructions when it comes to actual execution time.

```
r1 = r2 * r5
r3 = r1
r6 = r2 * r5
r4 = r6

r1 = …
r6 = …
…  = … r3
…  = … r4
```

_____

_____

r1 = …
r6 = …

What terms describe these particular kinds of peephole optimizations? List two that apply.

1)

2)

What C/C++ compiler option should you use to produce a `.o` file from a .c file? _____

What C/C++ compiler option should you use to produce a `.s` file from a .c file? _____

12. Extra Credit

What gets printed when this C program is executed?

```
#include <stdio.h>

int
main()
{
  char a[] = "Mo Zach Theresa!";
  char *p1 = a + 7;
  char *p2 = a + 2;

  printf( "%c", *p2++  = *++p1 + 1 ); _____

  printf( "%c", *--p1 = *p2-- );      _____

  printf( "%c", ++*p1 + 1 );          _____

  printf( "%c", *++p2 = *p1 + 3);     _____

  printf( "%c", --*p2++ );            _____

  printf( "%c\n", *(a+3) = 1["job"] ); _____

  printf( "%d\n", p2 - a );           _____

  printf( "%s\n", a );                _____________________________________________

  return 0;
}
```

What is Rick's favorite bra size? _____

What gets printed if the following function is invoked as `recurse( 4, 10 )` ? Hint: Draw stack frames.

```
int
recurse( int a, int b )
{
  int local = a + b;
  int result;

  if ( local > 9 )
    result = local - recurse( a, b - 2 );
  else
    result = local;

  printf( "%d\n", result );

  return result;
}
```

Put answers here

Crossword Puzzle (next page) (1 point)

Hexadecimal - Character

00	NUL	01	SOH	02	STX	03	ETX	04	EOT	05	ENQ	06	ACK	07	BEL
08	BS	09	HT	0A	NL	0B	VT	0C	NP	0D	CR	0E	SO	0F	SI
10	DLE	11	DC1	12	DC2	13	DC3	14	DC4	15	NAK	16	SYN	17	ETB
18	CAN	19	EM	1A	SUB	1B	ESC	1C	FS	1D	GS	1E	RS	1F	US
20	SP	21	!	22	"	23	#	24	$	25	%	26	&	27	’
28	(	29	)	2A	*	2B	+	2C	,	2D	-	2E	.	2F	/
30	0	31	1	32	2	33	3	34	4	35	5	36	6	37	7
38	8	39	9	3A	:	3B	;	3C	<	3D	=	3E	>	3F	?
40	@	41	A	42	B	43	C	44	D	45	E	46	F	47	G
48	H	49	I	4A	J	4B	K	4C	L	4D	M	4E	N	4F	O
50	P	51	Q	52	R	53	S	54	T	55	U	56	V	57	W
58	X	59	Y	5A	Z	5B	[	5C	\	5D	]	5E	^	5F	_
60	‘	61	a	62	b	63	c	64	d	65	e	66	f	67	g
68	h	69	i	6A	j	6B	k	6C	l	6D	m	6E	n	6F	o
70	p	71	q	72	r	73	s	74	t	75	u	76	v	77	w
78	x	79	y	7A	z	7B	{	7C	|	7D	}	7E	~	7F	DEL

A portion of the Operator Precedence Table

Operator	Associativity
++ postfix increment	L to R
-- postfix decrement	
[] array element	
() function call	
* indirection	R to L
++ prefix increment	
-- prefix decrement	
& address-of	
sizeof size of type/object	
(type) type cast	
* multiplication	L to R
/ division	
% modulus	
+ addition	L to R
- subtraction	
.	
.	
.	
= assignment	R to L

Scratch Paper

Scratch Paper