Login Name _____ **Name _____**

Student ID _____ **Signature _____**

Final
CSE 131
Spring 2015

Page 1	**_____**	**(28 points)**
Page 2	**_____**	**(28 points)**
Page 3	**_____**	**(24 points)**
Page 4	**_____**	**(26 points)**
Page 5	**_____**	**(20 points)**
Page 6	**_____**	**(28 points)**
Page 7	**_____**	**(30 points)**
Page 8	**_____**	**(28 points)**
Subtotal	**_____**	**(212 points) = 100%**
Page 9 Extra Credit	**_____**	**(18 points) [8.5% 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 T_ASSIGN {: System.out.println("A"); :} Expr {: System.out.println("B"); :}
     |     Expr1 {: System.out.println("C"); :}
     ;

Expr1  ::= Expr1 T_PLUS {: System.out.println("D"); :} Expr2 {: System.out.println("E"); :}
     |     Expr2 {: System.out.println("F"); :}
     ;

Expr2  ::= Expr2 T_STAR {: System.out.println("G"); :} Des {: System.out.println("H"); :}
     |     Des {: System.out.println("I"); :}
     ;

Des  ::=   T_STAR {: System.out.println("J"); :} Des {: System.out.println("K"); :}
     |     T_PLUSPLUS {: System.out.println("L"); :} Des {: System.out.println("M"); :}
     |     T_ID
     ;
```

What is the output when parsing the follow expression (you should have 21 lines/letters in your output): [21pts]

```
e = a + ++b * *c + **d
```

Output

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

If variable `e` is defined to be type `int`, fill in the correct types for variables `a`, `b`, `c`, and `d` for this expression to be semantically correct [4pts]

```
_____ a = 5;

_____ b = 2;

_____ c = &a;

_____ d = &c;

    int       e;
```

What is the resulting value that will be stored in variable `e` when the expression shown above is executed? [3pts]

_____

2. Given the C array declaration on a SPARC architecture

```
int a[2][3];
```

What is the `sizeof( a )`? _____ [2pts]

Mark with an **A** the memory location(s) where we would find the array element `a[1][2]`: [4pts]

a:



(Each box represents a byte in memory)

Given the following Reduced-C code

```
structdef S1 { int x; float y; };
structdef S2 { int x; float y; };
structdef S3 { float x; int y; };
S1 a;
S2 b;
S3 c;
S1 d;
S2 e;
S3 f;
```

identify each of the following statements as either: [18pts]

(A) No Error (B) Compile-Time Error

____ `a = b;`	____ `c = d;`	____ `c.x = c.y;`
____ `a = c;`	____ `c = e;`	____ `a = (S1) f;`
____ `a = d;`	____ `c = f;`	____ `a = * (S1*) &f;`
____ `a = e;`	____ `e = f;`	____ `* (S2*) &a = * (S2*) &f;`
____ `a = f;`	____ `a.x == b.x;`	____ `a.x = c.x = e.x;`
____ `d.x = f.y;`	____ `b.x = c.x;`	____ `c.x = a.y + b.y;`

Using the Right-Left rule (which follows the operator precedence rules) write the definition of a variable named *summer* that is a pointer to a pointer to a pointer to a function that takes an int as a single parameter and returns a pointer to an array of 7 elements where each element is a pointer to a double. [4pts]

_____;

3. What gets printed in the following Reduced-C program? [24pts]

```
structdef MS {
    int id;
    int count;
    MS(int id) {
        static int count;
        this.id = id;
        this.count = ++count;
        cout << "CTOR: " << this.id << ", " << this.count << endl;
    }
    ~MS() {
        cout << "DTOR: " << this.id << ", " << this.count << endl;
    }
};

MS s1 : (1);

function : void foo(int n) {
    static MS s2 : (2);
    MS s3 : (3);
    if ( n == 0 ) {
        return;
    }
    foo(n-1);
    MS s4 : (4);
}

function : void main() {
    MS s5 : (5);
    foo(1);
}
```

_____

_____

_____

_____

_____

_____

_____

_____

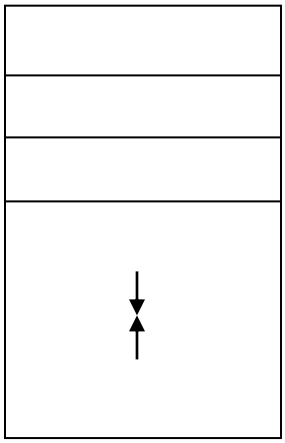
_____

_____

_____

_____

4. Fill in the names of the 5 main areas of the C/C++ Runtime Environment as laid out by most SPARC systems as described in class, and state what part(s) of a program are stored in each. [10pts]



low memory

_____

_____

_____

_____

_____

high memory

Fill in the body of the following Reduced-C function definition with a foreach loop that will update every element of array `myarr` with a value that is the square of the original element value: [8pts]

```
int myarr[10000];

function : void square_myarr() {




}
```

Given the following variables

```
int x = 17;       // initialized global variable
extern int y;     // external global variable
```

write the corresponding SPARC assembly code (including the section directive(s) and alignment) that should be generated to properly allocate any necessary space for the variables, and to ensure proper access/visibility to these variables if another file is linked to this code's object file. [8pts]

5. 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. Given the following Java class definitions, specify the output of each print() method invocation. [20pts]

```
class Copperhead {
    public void print(Copperhead viper) {
        System.out.println("A");
    }
}

class Cottonmouth extends Copperhead {
    public void print(Cottonmouth viper) {
        System.out.println("B");
    }
}

class BlackMamba extends Cottonmouth {
    public void print(Copperhead viper) {
        System.out.println("C");
    }
    public void print(BlackMamba viper) {
        System.out.println("D");
    }
}

public class DeadlyViperAssassinationSquad {
    public static void main (String [] args) {
       BlackMamba  viper1 = new BlackMamba();
       Cottonmouth viper2 = new BlackMamba();
       Cottonmouth viper3 = new Cottonmouth();
       Copperhead  viper4 = new BlackMamba();
       Copperhead  viper5 = new Cottonmouth();
       Copperhead  viper6 = new Copperhead();

       viper1.print( viper3 );                                  _____________________

       viper1.print( viper2 );                                  _____________________

       viper1.print( viper1 );                                  _____________________

       viper2.print( viper1 );                                  _____________________

       viper2.print( viper6 );                                  _____________________

       viper3.print( viper3 );                                  _____________________

       viper3.print( viper4 );                                  _____________________

       viper4.print( viper1 );                                  _____________________

       viper4.print( (Cottonmouth) viper1 );                    _____________________

       viper4.print( viper5 );                                  _____________________

       viper5.print( viper3 );                                  _____________________

       viper5.print( viper4 );                                  _____________________

       viper6.print( viper1 );                                  _____________________

       viper6.print( viper3 );                                  _____________________

       ( (BlackMamba) viper2 ).print( viper1 );                 _____________________

       ( (Cottonmouth) viper4 ).print( viper1 );                _____________________

       ( (Cottonmouth) viper4 ).print( viper2 );                _____________________

       ( (Cottonmouth) viper4 ).print( viper4 );                _____________________

       ( (Cottonmouth) viper5 ).print( viper3 );                _____________________

       ( (Cottonmouth) viper5 ).print( viper5 );                _____________________
    }
}
```

6. Given the following C++ program (whose semantics in this case is similar to our Reduced-C) and a real compiler's code gen (without optimization) as discussed in class, fill in the **values** stored in memory for each of the global and local variables and parameters in the run time environment for the SPARC architecture when the program reaches the comment `/* HERE */`. Do not add any unnecessary padding. [28pts]

```
int a = 10;

int factorial(int n) {
    static int b = n;

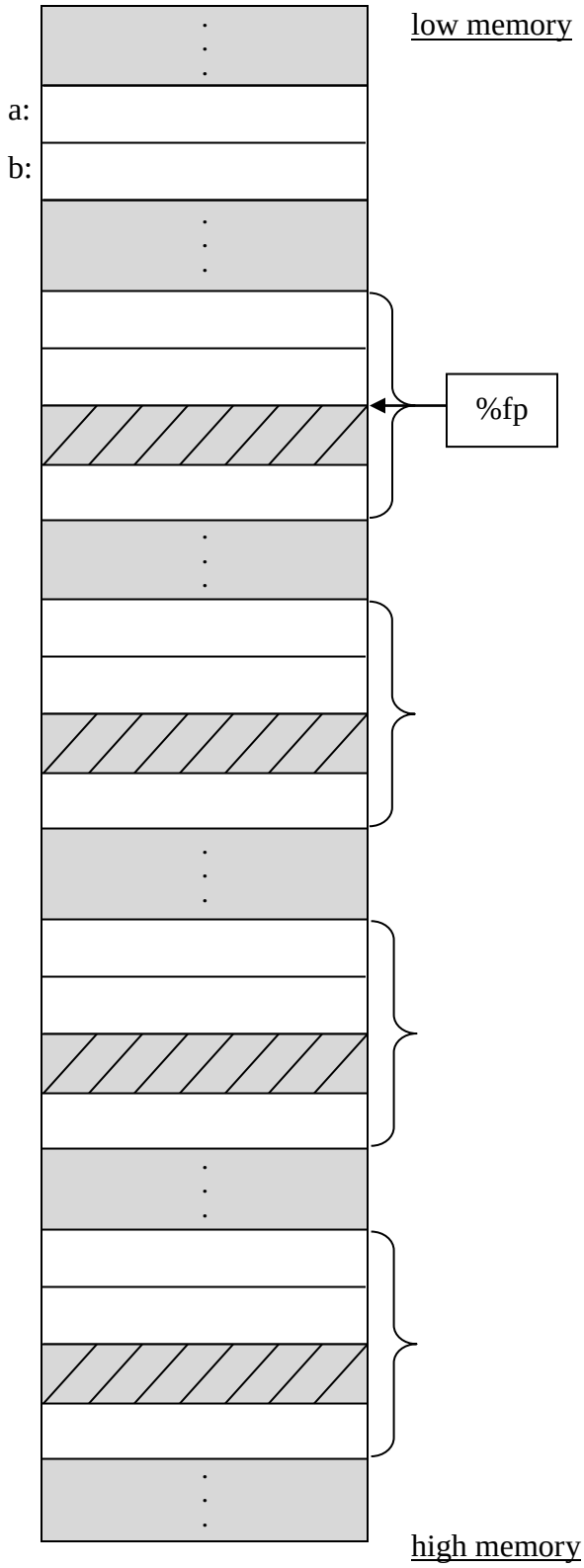
    int c[2] = {0, 0};

    c[ b++ % 2 ] = ++a;

    if ( n == 0 ) {
        /* HERE */
        return 1;
    }

    return n * factorial(n-1);
}

int main() {
    return factorial(3);
}
```



7. Given the following pseudocode, determine the program output based on the specified scoping rule [16pts]

```
int x;            -- global var declaration

void set_x( int n )
      x = n;

void alpha()
      set_x( 30 );
      cout << x;

void beta()
      int x = 50; -- local var declaration
      set_x( 40 );
      cout << x;

set_x( 20 );
alpha();
cout << x;
beta();
cout << x;
```

What does the program output if the language uses <u>static</u> scoping?

_____

_____

_____

_____

What does the program output if the language uses <u>dynamic</u> scoping?

_____

_____

_____

_____

From this quarter's Reduce-C spec (which follows closely the real C language standard), indicate whether each expression is either a: (A) Modifiable L-val (B) Non-Modifiable L-val (C) R-val [8pts]

```
function : int & foo() { /* function body not important. */ }
int x;
int * y = &x;
const int z = 5;
int a[z];
```

_____ foo() _____ ++foo() _____ &x _____ *y

_____ z _____ a _____ a[x] _____ a[*y]++

Identify the following C constructs as either: (A) Pure Declaration (B) Definition [6pts]

_____ extern int x; _____ int * foo(int *x) { return x; } _____ float * bar (float z);

_____ int a[50]; _____ extern int baz(int x, float y); _____ struct boo { int x; };

8. Given the following C type definitions: [18pts]

```
struct rock {
  short  a;
  double b;
  int    c;
};

struct paper {
  char        d;
  struct rock e;
  char        f;
};

struct scissors {
  short        g;
  struct paper h;
  char         i;
};

struct scissors roshambo;
```

Hint: Draw the memory layout of these structs (including padding) on the scratch paper at the end of this exam.

What is the `sizeof( struct rock )`? _____

What is the `sizeof( struct paper )`? _____

What is the `sizeof( struct scissors )`? _____

What is the `offsetof( struct paper, e.c )`? _____

What is the `sizeof( struct scissors )` if members of all 3 structs are reordered from largest alignment restriction to smallest? _____

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

Use virtual register notation for each of the following.

Change the following into a single instruction which is likely a time improvement over the current instructions when it comes to actual code generation. Assume `r7` is free and only `r4` is used later. [6pts]

```
r1 = 23
r2 = 9
r3 = r2 + r1
r3 = r3 + 0
r4 = r3 * r4
```

_____

What four terms describe these particular peephole optimizations?

1) _____

2) _____

3) _____

4) _____

Change the following into two instructions which are likely a time improvement over the current instructions when it comes to actual code generation. Assume `r7` is free and `r1` and `r2` are used later. [4pts]

```
r1 = 17 % 4
r2 = r1 % 2
```

_____

_____

What two terms describe these particular peephole optimizations?

1) _____

2) _____

9. Extra Credit

What gets printed when the following C program is executed? [18pts]

```
#include <stdio.h>

int main() {

  char a[] = "********";
  char *p = a;

  printf( "%c\n", *++p = 5[a] + 27 );                  ______

  printf( "%c\n", (p[3] = *p + 4) + 15 );              ______

  printf( "%c\n", (*p++ = 4[a]) + 11 );                ______

  printf( "%c\n", (*++p = a[1] + 1) - 5 );             ______

  printf( "%c\n", *++p + 9 );                          ______

  printf( "%c\n", p[-2] = p[1] = p[-1] + 4 );          ______

  printf( "%c\n", (p[2] = *p - 2) - 6 );               ______

  printf( "%c\n", p[-4] = ++p[-1] + 1 );               ______

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

  return 0;

}
```

A portion of the C Operator Precedence Table

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

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

Scratch Paper

Scratch Paper