Login Name _____ **Name _____**

Student ID _____ **Signature _____**

Final
CSE 131
Fall 2015

Page 1	**_____**	**(26 points)**
Page 2	**_____**	**(24 points)**
Page 3	**_____**	**(22 points)**
Page 4	**_____**	**(18 points)**
Page 5	**_____**	**(20 points)**
Page 6	**_____**	**(22 points)**
Page 7	**_____**	**(24 points)**
Page 8	**_____**	**(32 points)**
Subtotal	**_____**	**(188 points) = 100%**
Page 9 Extra Credit	**_____**	**(18 points) [9.6% 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 ::=   Expr T_MINUS {: System.out.println("A"); :} Des {: System.out.println("B"); :}
     |     Des {: System.out.println("C"); :}
     ;

Des  ::=   T_STAR {: System.out.println("D"); :} Des {: System.out.println("E"); :}
     |     T_AMPERSAND {: System.out.println("F"); :} Des {: System.out.println("G"); :}
     |     T_MINUSMINUS {: System.out.println("H"); :} Des {: System.out.println("I"); :}
     |     Des2
     ;

Des2 ::=   Des2 {: System.out.println("J"); :} T_MINUSMINUS {: System.out.println("K"); :}
     |     T_ID
     ;
```

What is the output when parsing the follow expression (you should have 19 lines/letters in your output): [19pts]

```
a - --*b-- - ***&c
```

Output

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

If the resulting type of the above expression must be an `int`, fill in the correct types for variables `a`, `b`, and `c` for this expression to be semantically correct [3pts]

_____ a;

_____ b;

_____ c;

Consider instead if the program had the following variables defined:

```
auto u = -10;
auto w = 20;
auto x = 30;
auto y = &x;
auto z = &y;
auto a = w;
auto b = &u;
auto c = z;
```

What is the resulting value when the original expression is computed? [4pts]

_____

2. What gets printed in the following Reduced-C program based on this quarter's spec and lecture? [24pts]

```
structdef MS {
    int id;
    MS(int id) {
        this.id = id;
        cout << "C: " << this.id << endl;
    }
    ~MS() {
        cout << "D: " << this.id << endl;
    }
};

MS a : (9);
static MS * b;

function : void bar(int n) {
    new b : (n--);
    MS d : (--n);
}

function : void foo(int n) {
    new b : (n--);
    static MS c : (n);
    MS d : (--n);
    bar(n-1);
}

function : void main() {
    MS e : (0);
    foo(6);
}
```

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

3. 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. [22pts]

```
int x;
auto y = 1.23;

void bob(int a) {
    a = a + 100;
    int b = 1;
    auto c = &y;

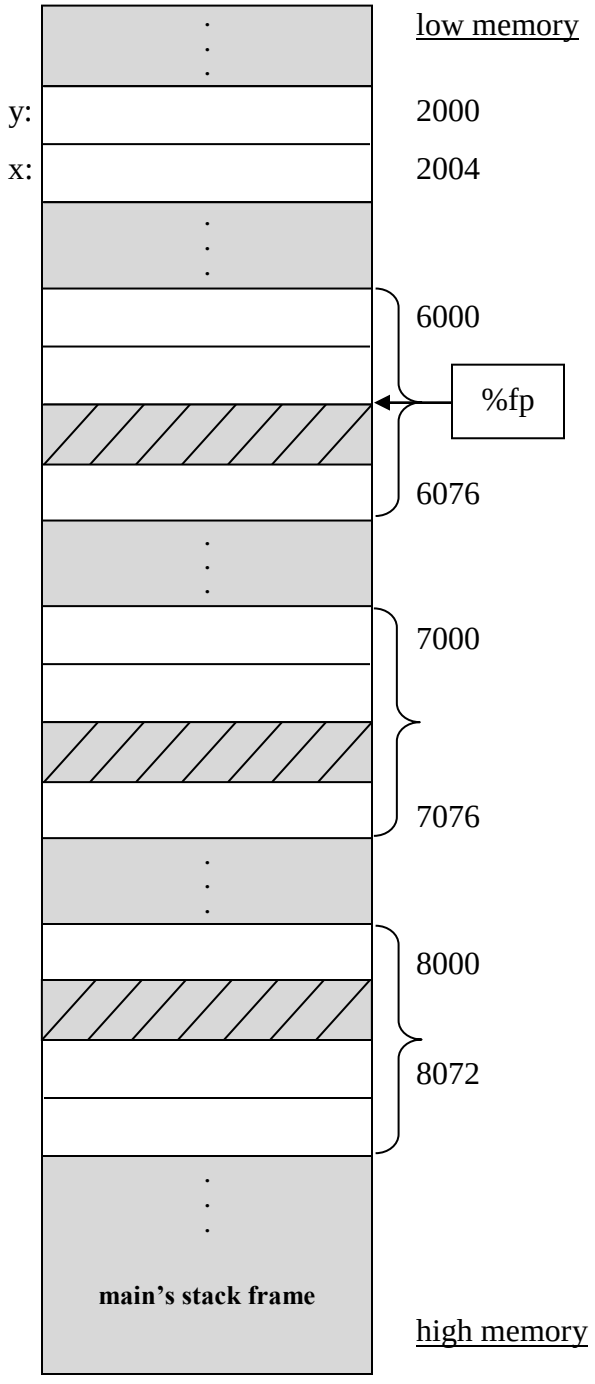
    /* HERE */
}

void stuart(float & a) {
    a = a + 5;
    int b = 1;
    int * c = &b;
    bob(b);
}

void kevin(int a, float & b) {
    a = a + 10;
    b = b + 20;
    float c = a + b;
    stuart(c);
}

int main() {
    kevin(x, y);
    return 0;
}
```

hypothetical ***decimal*** memory locations



4. 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. [18pts]

```
class Parallelogram {
    public void print(Parallelogram shape) {
        System.out.println("A");
    }
    public void print(Square shape) {
        System.out.println("B");
    }
}

class Rectangle extends Parallelogram {
    public void print(Parallelogram shape) {
        System.out.println("C");
    }
}

class Square extends Rectangle {
    public void print(Parallelogram shape) {
        System.out.println("D");
    }
    public void print(Rectangle shape) {
        System.out.println("E");
    }
}

public class GeometricQuandry {
    public static void main (String [] args) {
       Parallelogram shape1 = new Parallelogram();
       Parallelogram shape2 = new Rectangle();
       Parallelogram shape3 = new Square();
       Rectangle     shape4 = new Rectangle();
       Rectangle     shape5 = new Square();
       Square        shape6 = new Square();

       shape1.print( shape6 );                                  _____

       shape1.print( shape4 );                                  _____

       shape1.print( shape2 );                                  _____

       shape2.print( shape6 );                                  _____

       shape2.print( shape4 );                                  _____

       shape2.print( shape2 );                                  _____

       shape3.print( shape6 );                                  _____

       shape3.print( shape5 );                                  _____

       shape3.print( shape3 );                                  _____

       shape4.print( shape5 );                                  _____

       shape4.print( shape6 );                                  _____

       shape5.print( shape5 );                                  _____

       shape5.print( shape6 );                                  _____

       shape6.print( shape6 );                                  _____

       shape6.print( shape4 );                                  _____

       shape6.print( shape2 );                                  _____

       ( (Rectangle) shape2 ).print( shape4 );                  _____

       ( (Square) shape3 ).print( shape4 );                     _____
    }
}
```

5. Given the following pseudocode, determine the program output based on the specified scoping rule: [20pts]

```
int yin;                    // global var declaration
int yang;                   // global var declaration

void twiddle() {
    yin -= 2;
    yang += 2;
}

void frobnicate() {
    twiddle();
    if ( yin < yang ) {
        yin += 8;
        yang -= 8;
    }
}

void beta() {
    int yin = yang + 2;     // local var declaration
    frobnicate();
}

void alpha() {
    int yang = yin - 3;     // local var declaration
    beta();
    twiddle();
}

void init( int value ) {
    yin = value;
    yang = -value;
    twiddle();
    twiddle();
    if ( yin < yang ) {
        alpha();
        cout << yin << endl;
    } else {
        beta();
        cout << yang << endl;
    }
}
```

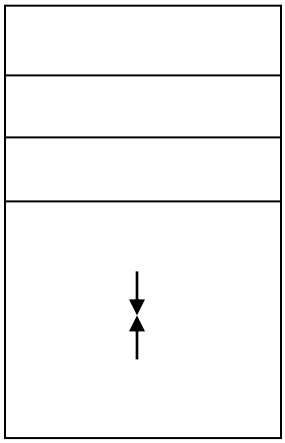
What does the program output if the language uses **<u>static</u>** scoping for each of the following function calls to `init()`?

```
init(-1);     // _____
init(2);      // _____
init(0);      // _____
init(-9);     // _____
init(3);      // _____
```

What does the program output if the language uses **<u>dynamic</u>** scoping for each of the following function calls to `init()`?

```
init(-2);     // _____
init(7);      // _____
init(1);      // _____
init(5);      // _____
init(-10);    // _____
```

6. 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

For the following Reduced-C program, indicate whether each line causes an illegal overload definition or not:
(A) No error (B) Illegal Overload Definition [6pts]

```
function : void xyz(       ) { /* valid code */ }          // ________

function : int &xyz(       ) { /* valid code */ }          // ________

function : int &xyz(int   x) { /* valid code */ }          // ________

function : void xyz(bool  x) { /* valid code */ }          // ________

function : bool xyz(int & x) { /* valid code */ }          // ________

function : void xyz(int   x) { /* valid code */ }          // ________
```

For the following Reduced-C program, indicate which overloaded function # will be bound at compile-time. If the function call is an illegal overloaded call, indicate the **number as 0**. [6pts]

```
function : int  foo(int x)          { /* valid code */ }          // #1
function : bool foo(float x, int y) { /* valid code */ }          // #2
function : bool foo(int x, float y) { /* valid code */ }          // #3
function : bool foo(float x)        { /* valid code */ }          // #4
function : int& foo(bool & x)       { /* valid code */ }          // #5

function : void main() {
      foo(4.4);               // ________

      foo();                  // ________

      foo(1, 1.5);            // ________

      foo(1 == 1);            // ________

      foo(4, 4);              // ________

      foo((int)3.33);         // ________
}
```

7. Given the following assembly code, draw a box around each basic block and label each box with a consecutive basic block number (e.g. #1, #2, #3, etc.): [24pts]

```
    set         x, %o0
    add         %g0, %o0, %o0
    set         4, %o1
    sub         %o0, %o1, %o0
    set         -8, %o1
    add         %fp, %o1, %o1
    st          %o0, [%o1]
.$$.loopCheck.1:
    set         -8, %o1
    add         %fp, %o1, %o1
    ld          [%o1], %o0
    set         4, %o2
    add         %o0, %o2, %o0
    st          %o0, [%o1]
    set         x, %o0
    add         %g0, %o0, %o0
    set         160, %o1
    add         %o0, %o1, %o1
    set         -8, %o0
    add         %fp, %o0, %o0
    ld          [%o0], %o0
    cmp         %o0, %o1
    bge         .$$.loopEnd.1
    nop
    set         -4, %o1
    add         %fp, %o1, %o1
    ld          [%o0], %o0
    st          %o0, [%o1]
    set         5, %o1
    cmp         %o0, %o1
    bne         .$$.cmp.1
    mov         %g0, %o0
    inc         %o0
.$$.cmp.1:
    set         -12, %o1
    add         %fp, %o1, %o1
    st          %o0, [%o1]
    set         -12, %l7
    add         %fp, %l7, %l7
    ld          [%l7], %o0
    cmp         %o0, %g0
    be          .$$.else.1
    nop
    ba          .$$.loopCheck.1
    nop
.$$.else.1:
    set         -4, %l7
    add         %fp, %l7, %l7
    ld          [%l7], %o0
    set         1, %o1
    add         %o0, %o1, %o0
    set         -16, %o1
    add         %fp, %o1, %o1
    st          %o0, [%o1]
    set         -4, %o1
    add         %fp, %o1, %o1
    set         -16, %l7
    add         %fp, %l7, %l7
    ld          [%l7], %o0
    st          %o0, [%o1]
    ba          .$$.loopCheck.1
    nop
.$$.loopEnd.1:
    ret
    restore
```

8. Given the following C type definition: [16pts]

```
struct happy {
  char     h;
  short    o;
  int      l;
  float    i;
  double   d;
  float  * a[1];
  int    * y[1];
  char     s[2];
};
```

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

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

What is the `offsetof( struct happy, a[0] )`? _____

What is the optimized `sizeof( struct happy )` if the member fields are reordered appropriately? _____

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

Give the order of the general phases of compilation in a typical compiler as discussed in class: [4pts]

A – Machine-independent code improvement (optional)
B – Machine-specific code improvement (optional)
C – Parser (Semantic analysis/intermediate code gen.)
D – Parser (Syntax analysis)
E – Scanner (Lexical analysis)
F – Source language (e.g. C)
G – Target code generation
H – Target language (e.g. assembly)

_____ –> _____ –> _____ –> _____ –> _____ –> _____ –> _____ –> _____

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 [12pts]

```
function : int a() { /* function body not important. */ }
const int b = 0;
float c = b;
float * d = &c;
bool e[10];
```

_____ `a()`	_____ `b`	_____ `&b`	_____ `*&b`
_____ `c`	_____ `c++`	_____ `d`	_____ `d[b]`
_____ `++*d`	_____ `e`	_____ `e[a()]`	_____ `!e[b]`

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)[2] = a[1] + 41 );            ______

  printf( "%c\n", (*p++ = 3[a] - 3) + 2 );           ______

  printf( "%c\n", ((++p)[4] = *a + 13) + 10 );       ______

  printf( "%c\n", (*++p = *(a+1) - 1) + 8 );         ______

  printf( "%c\n", (*a = 2[p] = a[7] + 14) + 13 );    ______

  printf( "%c\n", (*++p = a[6]) - 4 );               ______

  printf( "%c\n", (p[-3] = *p + 4) + 11 );           ______

  printf( "%c\n", --*p + 15 );                       ______

  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