Student ID _____ **Name _____**

Login Name _____ **Signature _____**

Final
CSE 131
Spring 2010

Page 1	**_____ (25 points)**
Page 2	**_____ (14 points)**
Page 3	**_____ (20 points)**
Page 4	**_____ (27 points)**
Page 5	**_____ (23 points)**
Page 6	**_____ (22 points)**
Page 7	**_____ (13 points)**
Page 8	**_____ (19 points)**
Page 9	**_____ (34 points)**
Page 10	**_____ (15 points)**
Page 11	**_____ (13 points)**
Subtotal	**_____(225 points = 100%)**
Page 12 Extra Credit	**_____ (13 points) [over 5% EC]**
Total	**_____**

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

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

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

Des2 ::=   Des2 {: System.out.println("J"); :} T_PLUSPLUS {: System.out.println("K"); :}
     |     Des3 {: System.out.println("L"); :}
     ;

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

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

What is the output when parsing the follow expression (you should have 22 lines/numbers in your output):

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

Output

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

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

_____ x; _____ z;

Which op in the Des production has higher precedence? Put an X in front of your answer.

___ T_STAR ___ T_PLUSPLUS ___ T_AMPERSAND ___ All same precedence

2. Pointer arithmetic is scaled based on the size of the data type the pointer is defined to point to. Based on the following global variable definitions, identify which C code fragment(s) is/are correct and which is/are incorrect and briefly explain why in all cases. The code fragments should set a[0] to 42 and a[1] to 24.

```
int a[2];
int * b;

b = a;
*b = 42;
b = b + 1;
*b = 24;
```

State whether correct/incorrect and why.

```
b = a;
*b = 42;
b = b + 1 * sizeof( int );
*b = 24;
```

```
b = a;
*b = 42;
b = b + 1 * sizeof( int * );
*b = 24;
```

```
b = a;
*b = 42;
b = b + 1 * sizeof( a );
*b = 24;
```

If we put the keyword `extern` in front of the variable definitions of `a` and `b` above (to make them pure declarations) and separately compile this source file to a `.s` file, would the compiler still be able to generate the proper code for the correct code fragment(s) above? State Yes or No and explain why.

What compiler option should you give to produce a `.s` file? _____
What compiler option should you give to produce a `.o` file? _____

Explain the main difference(s) that the compiler does/doesn't do in code gen for the case where we put the keyword `extern` in front of the variable definitions of `a` and `b` above compared to the original global variable definitions initially specified. Be specific. If there is more than one thing that is different, label them #1, #2, ...

3. In object-oriented languages like Java, determining which overloaded method code to bind to (to execute) is done at run time rather than at compile time (this is known as dynamic dispatching 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.

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

```
class Moe extends Curly {
    public void print(Moe m) {
        System.out.println("Moe 1");
    }

    public void print(Curly c) {
        System.out.println("Moe 2");
    }
}
```

```
class Larry extends Moe {
    public void print(Larry l) {
        System.out.println("Larry 1");
    }

    public void print(Curly c) {
        System.out.println("Larry 2");
    }

    public void print(Moe m) {
        System.out.println("Larry 3");
    }
}
```

Now <u>remove</u> the entire method `print(Moe m){}` in `class Moe` and <u>remove</u> the entire method `print(Moe m){}` in `class Larry`. Specify the output of each print() method <u>with these changes</u> below.

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

        Curly stooge1 = new Curly();
        Curly stooge2 = new Moe();
        Curly stooge3 = new Larry();
        Moe   stooge4 = new Moe();
        Moe   stooge5 = new Larry();
        Larry stooge6 = new Larry();

        stooge1.print( stooge1 );

        stooge2.print( stooge2 );

        stooge3.print( stooge3 );

        stooge4.print( stooge4 );

        stooge5.print( stooge5 );

        stooge6.print( stooge6 );
    }
}
```

_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Match the part of the calling convention with the tasks performed.

A) Pre-Call (Caller) B) call/jsr C) Post-Call (Caller) D) Prologue (Callee) E) Epilogue (Callee)

_____ Copies actual arguments into argument space

_____ Retrieves return value from return value location

_____ Stores return value into return value location

_____ Allocates space for local variables

_____ Saves registers in callee-save scheme

_____ Saves %pc into the return address location

_____ Retrieves saved return address for return/rts

_____ Performs initialization of local variables

4. With regards to codegen in Project 2, explain why it is not enough to store just the *value* for the ExprSTO resulting from a pointer dereference using the unary dereference operator (*). In your explanation, clearly indicate what other information needs to be stored for the resulting ExprSTO, and include a specific example showing why this needs to be done.

Using Reduced-C syntax, first define a struct A with members of type bool and pointer to struct A named x and ptr, respectively. Then define a variable named `foo` which is an array of an array (with dimensions 4x9) of pointers to struct A such that `foo[0][0]->ptr = foo[3][8];` is a valid expression. This will take more than one line of Reduced-C code. You can use two columns in the space below if you wish: struct definition on the left, array definition on the right.

For each of the following make no assumptions of what may be above or below each window of instructions unless otherwise stated. Use virtual register notation (not assembly language). Loads/Stores between reg/mem.

Change the following into two instructions which are most likely a time improvement over the original instructions when it comes to actual code execution. `x` represents a memory location.

```
r5 = 4 * 5
x = r5
r5 = x
r7 = r5 + 4
x = r3
r5 = r2 + r7
! r7 is dead from here on
```

What 4 terms describe these 4 particular kinds of peephole transformations?

1)

2)

3)

4)

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

```
r1 = r2 * 18
```

What term describes this particular kind of peephole optimization?

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!

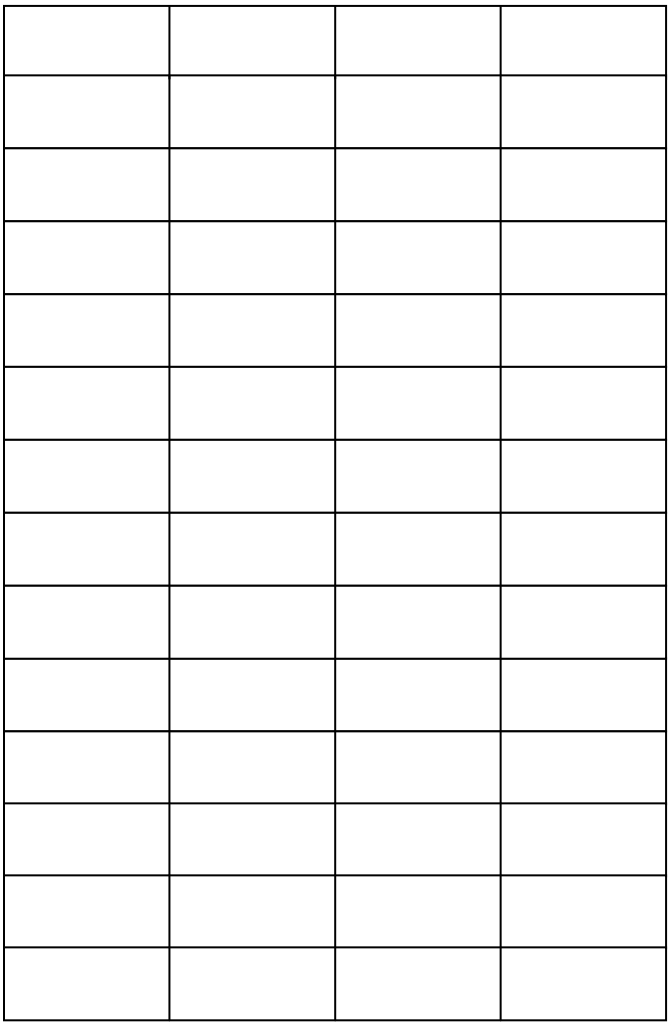
```
int a = 12;
int b = 45;
int c = 78;

void fubar( int x, int & y, int * z )
{
   x = x + 3;
   y = y + 3;
   *z = *z + 3;
}

void foo1( int & d, int * e, int f )
{
   d = d + 2;
   *e = *e + 2;
   f = f + 2;

   cout << a << endl;   ______
   cout << b << endl;   ______
   cout << c << endl;   ______
   cout << d << endl;   ______
   cout << *e << endl;  ______
   cout << f << endl;   ______

   fubar( d, d, &d );
   fubar( *e, *e, e );
   fubar( f, f, &f );

   cout << a << endl;   ______
   cout << b << endl;   ______
   cout << c << endl;   ______
   cout << d << endl;   ______
   cout << *e << endl;  ______
   cout << f << endl;   ______
}

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

   cout << a << endl;   ______
   cout << b << endl;   ______
   cout << c << endl;   ______

   return 0;
}
```

Using the Right-Left rule write the C definition of a variable named fubar that is a pointer to a pointer to a function that takes a pointer to a double and a pointer to a char and returns a pointer to an array of 7 elements where each element is a pointer to a 2-d array of 4 rows and 11 columns where each element is a pointer to a struct Sporcle.

6. Show the memory layout of the following C struct definition taking into consideration the **SPARC** data type memory alignment restrictions discussed in class. Circle this sentence for a point. Fill bytes in memory with the appropriate struct member/field name. For example, if member/field name `p` takes 4 bytes, you will have 4 `p`'s in the appropriate memory locations. If the member/field is an array, use the name followed by the index number. For example, some number of `p[0]`s, `p[1]`s, `p[2]`s, etc. If the member/field is a struct, use the member name followed by its member names (e.g. p.a, p.b). Place an `X` in any bytes of padding. Structs and unions are padded so the total size is evenly divisible by the most strict alignment requirement of its members. Use your scratch paper if you need to fill in actual values in the memory locations.

```
struct foo {
  char   a[3];
  short  b;
  char   c;
  double d;
  int    e;
};

struct fubar {
  double *   f;
  char       g[9];
  struct foo h;
  char       i[1];
  short      j;
};

struct fubar fubaz;

fubaz.f = &fubaz.h.d;
strcpy(fubaz.h.a, "AB");
strcpy(fubaz.g, "Hello2Me");
fubaz.i[0] = 'J';
fubaz.h.e = 0x31323334;
fubaz.j = 0x4142;
fubaz.h.d = 0x0;
fubaz.h.b = fubaz.j;
fubaz.h.c = 'K';
```

low memory
fubaz:



high memory

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

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

What is the hexadecimal value at the memory location referred to by the following expressions?

```
    ( (struct fubar *) (&fubaz.h.d) ) -> g[4]          _____

*(( (char *) (& ((&fubaz.h) -> d) + 1) ) + 1)          _____

*(char *) ((short *) ( ((int *) fubaz.f) + 2) + 1)     _____
```

7. Given the C array declaration

```
          C
short a[3][4];
```

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

```
      a[1][1]
a:
```

```
low memory                                                    high memory
```

Each box above represents a byte in memory.

Why would main() need an initialization guard in our Reduced-C implementation? Be specific.

With respect to the assembly code generated, what is different between taking the address of a global variable and taking the address of a local variable? Be specific.

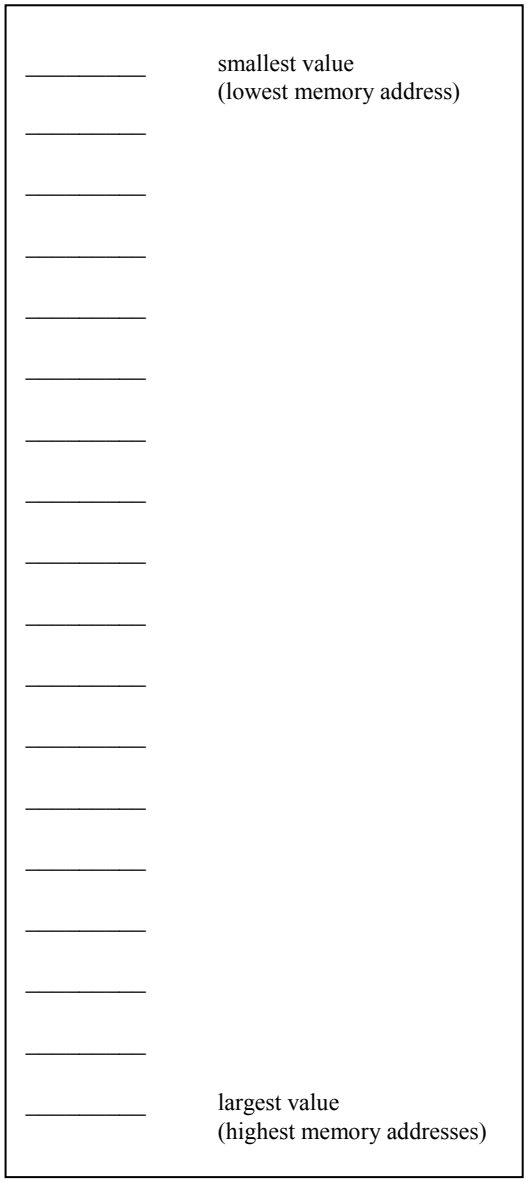
In this quarter's project we stated that if an extern function (function prototype) or extern variable is declared, we would not define it in the same file (have its full definition in the same source file). Suppose we want to remove this restriction, so that it is legal to define an extern function or variable in the same source file. What change(s) would need to be made in order to support this (allow an extern declaration say at the top of the file and then have the definition later in the same source file)? Be specific.

What is the output of the following C++ program (similar to this quarter's Reduced-C spec)?

```
void foo( int x )
{
   static int y = x - 2;

   cout << ++y << endl;

   if ( x >= 2 && y <= 5 )
       foo( y++ - 2 );
   else
       cout << "Stop!" << endl;

   cout << ++y << endl;
}

int main()
{
  foo( 5 );
  return 0;
}
```

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 */

int a = 37;

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

   int b = 42;
   int c;

   foo1( &argc, a );

/*  1 */ (void) printf( "1: argv --> %p\n", &argv );
/*  2 */ (void) printf( "2: foo1 --> %p\n", foo1 );
/*  3 */ (void) printf( "3: b --> %p\n", &b );
/*  4 */ (void) printf( "4: c --> %p\n", &c );
/*  5 */ (void) printf( "5: argc --> %p\n", &argc );
}

void foo1( int *c, int d ) {

   int e;
   struct foo {int a; int b;} f;

   foo2( e, c );

/*  6 */ (void) printf( "6: c --> %p\n", &c );
/*  7 */ (void) printf( "7: e --> %p\n", &e );
/*  8 */ (void) printf( "8: f.a --> %p\n", &f.a );
/*  9 */ (void) printf( "9: malloc-1 --> %p\n",
                        c = (int *) malloc(150) );
/* 10 */ (void) printf( "10: d --> %p\n", &d );
/* 11 */ (void) printf( "11: a --> %p\n", &a );
/* 12 */ (void) printf( "12: f.b --> %p\n", &f.b );
}

void foo2( int h, int *i ) {

   static int j;
   int k[2];

/* 13 */ (void) printf( "13: i --> %p\n", &i );
/* 14 */ (void) printf( "14: k[1] --> %p\n", &k[1] );
/* 15 */ (void) printf( "15: malloc-2 --> %p\n",
                          i = (int *) malloc(50) );
/* 16 */ (void) printf( "16: h --> %p\n", &h );
/* 17 */ (void) printf( "17: j --> %p\n", &j );
/* 18 */ (void) printf( "18: k[0] --> %p\n", &k[0] );
}
```



Variables declared to be _____ will not be optimized by the compiler.

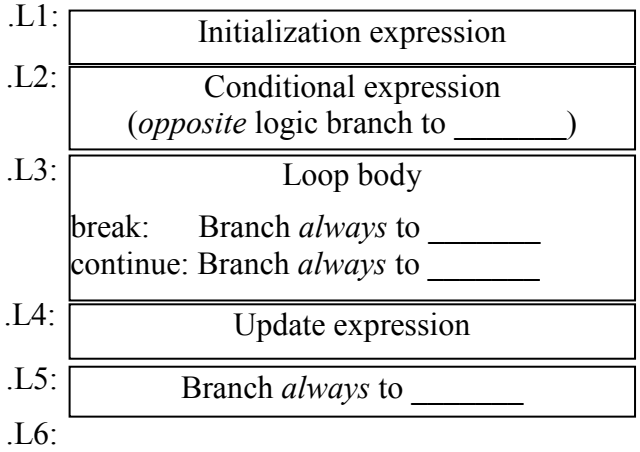
9. Given the following Reduced-C code fragment:

```
function : void main(){
    int a = 2;  //Assume a is located at %fp - 4

    /* code that may change a */

    cout << a + 3.0 << endl;
}
```

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 _____________

ENDL:   _______________ "\n"                ! newline string

        .section _____________
        .align 4

        _____________ main
main:
        set ____________________, %g1

        ________ ________________, %g1, ________________

        ________ ________, %l0              ! initialize local variable a to the value 2

        ________ %l0, [________________]

   /* Code that may change a */

        .section _____________
        .align 4

FLOAT1: ________________ 0r3.0              ! store floating point constant value 3.0 for later use

        .section _____________
        .align 4

        ld [_____________], %f0             ! get value of local var a for FP conversion

        _____________ %f0, %f0              ! convert value of a to single precision FP

        st  %f0, [_____________]            ! store converted value in temp1


        ld  [_____________], %f0            ! get converted value from temp1, put in %f0

        set _____________, %l1              ! get floating point constant 3.0, put in %f1

        ________ [%l1], %f1

        ________ %f0, %f1, %f2              ! a + 3.0

        ________ %f2, [%fp + -12]           ! save result in temp2


        ld [%fp + -12], ________

        call ____________________           ! print result of (a + 3.0)

        ________


        set _____________, %o0

        call _____________                  ! print the newline string

        ________


        ________

        _______________

main.SIZE = -(92 + ______) ______ ________  ! save space for 1 local var and 2 temporaries
```

10. The following is an assembly code structure for one way of implementing for-loops in this quarter's project. Fill in the blanks to indicate which of the labels each branch should be taken to.



Give an example of something (either in C/C++ or our Reduced-C) that is:

a) an r-value (neither addressable nor assignable)

b) a non-modifiable l-value (an object locator that is addressable but not assignable).

c) a modifiable l-value (an object locator that is addressable and assignable)

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

A) Loader B) Linkage Editor C) C Compiler D) Assembler E) C Preprocessor

_____ combines all object modules into a single executable file.

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

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

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

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

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

_____ translates C code into assembly code.

_____ translates assembly code into machine code.

11. Assume you have the following declarations:

```
int a[3][5];
int (*b)[5];
int * c[3];
int ** d;
int i, x, y;
```

Complete the following code to initialize `b` so that the addresses `&a[x][y]` and `&b[x][y]` are always equivalent to each other:

```
b = _____________________;
```

Complete the following code to initialize `c` so that the addresses `&a[x][y]` and `&c[x][y]` are always equivalent to each other:

```
for(i = 0; i < 3; ++i)
    c[ _____ ] = _____________________;
```

Complete the following code to initialize `d` so that the addresses `&a[x][y]` and `&d[x][y]` are always equivalent to each other:

```
d = malloc( _____________________ );

for(i = 0; i < 3; ++i)
    d[ _____ ] = _____________________;
```

Give the values for the following:

```
sizeof(a): _________          sizeof(c): _________

sizeof(b): _________          sizeof(d): _________
```

Write an equivalent expression for `d[x][y]` without using any array indexing brackets.

Define and initialize a traversal pointer named `ptr` for efficiently traversing every element of the array `a`.

Use that traversal pointer to write an expression equivalent to the following expression: `a[x][y]`

12. Extra Credit

What gets printed when this program is executed?

```
#include <stdio.h>

int
main()
{
  char a[] = "diversion";
  char b[] = "sound";
  char *p1 = a;
  char *p2 = b;

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

  printf( "%c", a[strlen(a)-1] - 2 );   _____

  printf( "%c", *(b + 4) - 1 );         _____

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

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

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

  printf( "%c", p2[-1] + 1 );           _____

  /* BTW, Diversion Sound is Kellen Steffen's band. */

  printf( "%c", p1[2] + 3 );            _____

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

  printf( "%c", ++*p1 + 2 );            _____

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

  return 0;
}
```

What is the question?

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 C 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