1. Using the operator precedence table above, evaluate each expression and state what gets printed.

```c
int x = 3;
int a = 16;
int b = 11;
x = a + b % x * x - b;
printf( "%d\n", x );
```

Value: 18

```c
int x = 3;
int a = 16;
int b = 11;
x = b + x - b * x / a;
printf( "%d\n", x );
```

Value: 10

2. What gets printed in the following blocks of statements?

```c
int a = 4;
int b = 5;
int c = -7;
if ( (a < 6) && !(b < 8) || !(c != a) )
    printf( "True" );
else
    printf( "False" );
```

No printout

```c
int x = -3;
int y = 10;
int z = x + 9;
if ( (z == 8) || !(x > y) && (y >= z) )
    printf( "True" );
else
    printf( "False" );
```

No printout
3. Fill in the blanks for the appropriate compilation sequence. (6 pts)

A) Executable Program   D) C Compiler
B) Linker/Linkage Editor   E) C Preprocessor
C) Assembler     F) C Source Code

_________ —> _________ —> __________ —> _________ —> _________ —> ___________

4. Now match what various parts of the compilation sequence does. The letters above may be used more than once or not at all. (12 pts)

_____ Translates a program written in assembly language into an equivalent program represented in machine code (0's and 1's)
_____ Expands #include and #define statements
_____ Translates a program written in C language into an equivalent program represented in assembly language
_____ Strips away comments
_____ Brings all the object modules together including those from the Standard C Library used in your program and produces a file that can be executed
_____ Performs syntax (form/structure) and semantic (meaning) analysis of your program

5. Write a function called checkRange that checks if the first argument is between the second and third arguments inclusive. You can assume the second argument is less than or equal to the third argument. Return the integer value 1 to indicate YES (the first argument is between the second and third argument); return the integer value 0 to indicate NO (the first argument is not between the second and third argument).

Fill in the blanks to complete this function. (15 pts)

Examples:    checkRange( 10, 10, 20 ) would return 1    checkRange( 8, 10, 20 ) would return 0
            checkRange( 30, 20, 30 ) would return 1    checkRange( 43, -9, 33 ) would return 0
            checkRange( 25, 22, 44 ) would return 1    checkRange( 19, 19, 19 ) would return 1

________ checkRange( int value, int minValue, int maxValue )
{
    if ( _________________________________ && _________________________________ )
        return 1;

    else
        _________________ ;
}
6. Write an equivalent **switch** statement for the following **if-else** statement.  
(16 pts)

```c
if ( x == 5 || x == 18 )
{
    x = x + 23;
    printf( "%d", x );
}
else if ( x == -7 )
{
    x = 420 / x;
}
else
{
    printf( "Switchfoot" );
    x = 666 * 0;
}
```

7. What gets printed in the following block of statements?  
(8 pts)

```c
#define SIZE 8
int i;
int array[SIZE] = { -11, 2, 7, 4, 6, 3, 12, 5 };

for ( i = 0; i < SIZE; ++i )
    if ( (array[i] * 2) >= 10 )
        printf( "%d\n", array[i] );
```

8. What gets printed?  
(8 pts)

```c
#include <stdio.h>

int function1( float param1, int param2 );

void main( void )
{
    float i = 3.14;
    int j = 3;

    j = function1( i, j );
    printf( "%d\n", j );
}

int function1( float param1, int param2 )
{
    int i;

    for ( i = 1; i <= param2; ++i )
        printf( "%.2f\n", param1 + i );
    return (param2 * i);
}
9. What gets printed? (27 pts)

#include <stdio.h>

#define SIZE 7

int function2( int array[], int size );

void main( void )
{
    int array[SIZE] = { -2, 1, 4, 2, 3, -3, 6 }; int i, result;

    result = function2( array, SIZE );
    printf( "Returned value = %d\n", result );

    printf( "Array elements:\n" );
    for ( i = 0; i < SIZE; ++i )
        printf( "%d\n", array[i] );
}

int function2( int array[], int size )
{
    int i;
    int count = size;

    for ( i = size - 1; i > 0; --i )
    {
        if ( array[i] >= array[i-1] )
            array[i] = 2 * array[i-1];
        --count;
    }

    return count;
}
10. Consider the following program. Identify the marked parts, lifetime, and scope/visibility with the corresponding letter/digit from the lists below. (40 pts)

<table>
<thead>
<tr>
<th>C/C++ Program Part</th>
<th>Lifetime</th>
<th>Scope/Visibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>A) C Preprocessor Directive</td>
<td>1) Entire Program</td>
<td>WW) Just This Source Module</td>
</tr>
<tr>
<td>B) Global Variable</td>
<td>2) During func2() call</td>
<td>XX) Within func2() Only</td>
</tr>
<tr>
<td>C) Local Variable</td>
<td>3) During foo() call</td>
<td>YY) Entire Program</td>
</tr>
<tr>
<td>D) Function Definition</td>
<td></td>
<td>ZZ) Within foo() Only</td>
</tr>
<tr>
<td>E) Internal Static Variable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F) (Formal) Parameter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G) Function Prototype</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H) External Static Variable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```c
#include <stdio.h> ______
deﬁne SIZE 17 ______
int func2( char array[] ); ______ (entire line)
static long actor; ______ ______ ______
double ch = 4.20; ______ ______ ______
void foo( char ch )              (foo(){...}) ______ ______ ______
{
  (ch) ______ ______ ______
  static int actor; ______ ______ ______
  int result = 8;
  /* Other code here */
}
static int func2( char fubar[] )      (func2(){...}) ______ ______ ______
{
  (fubar) ______ ______ ______
  static int result = 19; ______ ______ ______
  char actor;
  /* Other code here */
}
```

How many times is the variable `result` in `func2()` initialized to 19 if `func2()` is called 6 times? _______ times

What is the initial value of the variable `actor` in `foo()`? ________

How many times is the variable `actor` in `foo()` given this value if `foo()` is called 6 times? _______ times

What is the initial value of the variable `actor` in `func2()`? ________

How many times is the variable `result` in `foo()` initialized if `foo()` is called 6 times? _______ times

Code in `foo()` that refers to the symbol/name `actor` refers to which symbol/name?

Code in `func2()` that refers to the symbol/name `actor` refers to which symbol/name?
11. Consider the following structure definition and variable declarations. (18 pts)

```c
struct Almost_Done
{
    float a;
    int b;
    float c[9];
    struct Almost_Done var1, var2, var3;
    int d[5];
    int e;
};
```

Fill in the blanks to complete the following tasks:

/* Read the value typed at the keyboard into the struct member b in var1 */
scanf( "%____\n", ________________ );

/* Print all elements of struct member c in var2 EXCEPT the first and last elements */
for ( i = ______ ; i < ______ ; ______ )
    printf( "%____\n", _________________ );

/* Assign the number of licks it takes to get to the center of a tootsie pop to the last element in struct member d in var3 */
____________________ = ____________________ ;

12. Consider the following strings variable definitions. (24 pts)

```c
char s1[] = " for an ";
char s2[] = "Eye";
char s3[40];
char s4[20] = "UNCLE";
char s5[] = "unknown";
strcpy( s3, s2 );
strcat( s3, s1 );
```

What gets printed?

printf( "%d", strlen( s3 ) );  _________
printf( "%d", sizeof( s1 ) );  _________

Fill in the blanks to complete the following tasks:

/* Change the 'C' in s4 to 'K' without using an explicit 'K' */
_____________________ = _______________________ ; /* CANNOT use 'K' */

/* Output "UNKLE - Eye for an Eye" in a single printf() statement. */
printf( "%___ %___ %___ %___", ___________, ____________, ___________, ___________ );