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

```c
int x;
int a = 15;
int b = 10;

x = a + b % 3 * 2 - b;
printf( "\%d\n", x );
```

(3 pts)

```c
int x;
int a = 15;
int b = 10;

x = b + 3 - 2 * b / a;
printf( "\%d\n", x );
```

(3 pts)

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

```c
int a = 8;
int b = 5;
int c = -7;

if ( !(a < 6) || (b > 8) || !(c == a) )
    printf( "True" );
else
    printf( "False" );
```

(3 pts)

```c
int x = -3;
int y = 10;
int z = x + 9;

if ( (z != 8) && !(x > y) )
    printf( "True" );
else
    printf( "False" );
```

(3 pts)
3. Which of the following are not valid C identifiers? (Circle your answer(s).)  (6 pts)

- 9_to_5
- FM94.9
- The_Pixies
- double
- string
- Char

4. Fill in the blanks for the appropriate compilation sequence. (12 pts)

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

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

5. What gets printed? (15 pts)

```c
void
main( void )
{
    int num = 6;
    switch ( num - 5 )
    {
    case 1:
        printf( "A\n" );
        num = num + 2;
    case 4:
        printf( "B\n" );
        num = num + 4;
        break;
    case 7:
        printf( "C\n" );
        num = num + num;
    case 6:
        printf( "D\n" );
        num = num + 5;
        break;
    default:
        printf( "E\n" );
        num = num + 8;
        break;
    }
    printf( "num = %d\n", num );
}
```

What would get printed if the switch statement read

switch( num - 4 ) instead of switch( num - 5 )?
6. Write an equivalent if-else for the following switch statement. (12 pts)

```c
switch ( value )
{
    case 3:
    case 6:
        value = value * 5;
        break;

    case 19:
        value = value + 420;
        break;
}
```

7. What gets printed in the following block of statements? (12 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) != 0 )
        printf( "%d\n", array[i] );
```

8. What gets printed? (8 pts)

```c
#include <stdio.h>
int function1( int var1, char var2 );

void main( void )
{
    int i = 3;
    char j = '2';

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

int function1( int var1, char var2 )
{
    int i;

    for ( i = 0; i < var1; ++i )
        printf( "%c\n", var2 );
    return (i + var1);
}
#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 = 0;

    for ( i = 1; i < size; ++i )
    {
        if ( array[i] >= array[i-1] )
        {
            array[i-1] = 2 * array[i];
            ++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) Entire Program</td>
</tr>
<tr>
<td>B) Global Variable</td>
<td>2) During foo() call</td>
<td>XX) This Source Module</td>
</tr>
<tr>
<td>C) Local Variable</td>
<td>3) During func2() call</td>
<td>YY) Within foo() only</td>
</tr>
<tr>
<td>D) Function Definition</td>
<td>ZZ) Within func2() only</td>
<td></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 5 ______
int func2( char array[] ); ______ (entire line)
long johns = 420; ______ ______ ______
static double ch; ______ ______ ______
static void
foo( char ch )              (foo(){...}) ______ ______ ______
{
  (ch) ______ ______ ______
  int array[SIZE] = { 0, 3, 5, 2, 6 }; ______ ______ ______
  static int result = 4;
  /* Other code here */
}
int
func2( char johns[] )      (func2(){...}) ______ ______ ______
{
  (johns) ______ ______ ______
  int i;
  static int result;
  /* Other code here */
}
```

How many times is the variable `result` in `foo()` initialized to 4 if `foo()` is called 8 times? _______ times

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

How many times is the variable `result` in `func2()` given this value if `func2()` is called 8 times? _______ times

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

How many times is the variable `array` in `foo()` initialized if `foo()` is called 8 times? _______ times

Code in `func2()` that refers to the symbol/name `ch` refers to which symbol/name?

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

```c
struct Almost_Done
{
    float a;
    int b;
    int c;
    char d[8];
    int e;
};

struct Almost_Done var1, var2, var3;
```

Fill in the blanks to complete the following tasks:

/* Print the value of the struct member c in var2 */

```c
printf( "%____\n", ______________ );
```

/* Print all the elements of the struct member d in var3 */

```c
for ( i = ______ ; i < _______ ; ______ )
    printf( "%____\n", ________________ );
```

/* Assign the value from struct member e in var2 to the struct member b in var1 */

```c
____________________ = ____________________;
```

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

```c
char s1[] = "Hello";
char s2[] = "Yellow?";
char s3[20];
strcpy( s3, s2 );
strcat( s3, s1 );
```

```c
What gets printed?
printf( "%d", strlen( s3 ) );  __________
printf( "%s", s3 );   __________________________________________
```

Fill in the blanks to complete the following tasks:

/* Output "Hello ? Yellow?" in a single printf() statement. */

```c
printf( "%___ %__ %__\n", ______________, ______________, ______________ );
```

/* Change s2 to be "Yellow!" instead of "Yellow?" */

```c
_____________________ = _______________________ ;
```

/* Change the 'H' in s3 to 'h' using the tolower() function */

```c
_____________________ = _______________________ ;
```