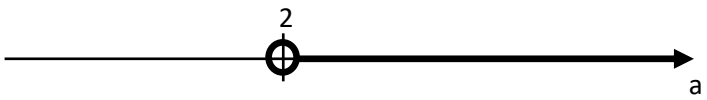
Nested Conditions and Logical Operators

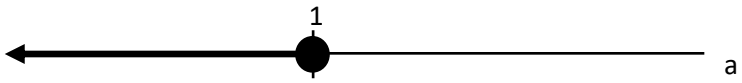
1. For each of the following Boolean expressions, show the values on the corresponding number line for which the expression evaluates to true(1). Recall the following operators: `&&` (logical **and**), `||` (logical **or**), `!` (logical **not**). The first two have been done for you…

a) `( a > 2 )`



Note: the open circle in the diagram above indicates that the number 2 is not included in the interval.

b) `( a <= 1 )`

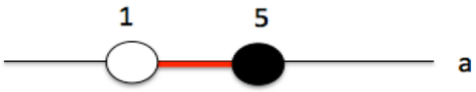


Note: the filled-in circle in the diagram above indicates that the number 1 is included in the interval.

c) `( a != 4 )`

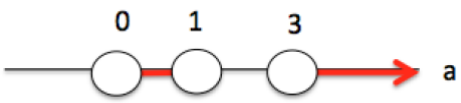
d) `( a < 2 || a > 4 )`

e) `( a > 1 && a <= 5 )`



f) `( a < 5 || a > 1 )`

g) `( ( a > 0 && a < 1 ) || a > 3 )`



2. What will happen when the following code snippet executes and why?

a)
```
if ( a < 5 || a > 1 )
        printf("Hello!\n");
```

The printf statement will execute no matter what the value of a is,
as this expression will always evaluate to 1 (True) for all values of a.

b)
```
if ( a > 5 && a < 1 )
        printf("Hello!\n");
```

The printf statement will not execute no matter what the value of a is,
as this expression will always evaluate to 0 (False) for all values of a.

3. In this question, assume that `data` is a variable of type `int`.

a) Write a Boolean expression that is true(1) when the value of `data` is less than 0 or bigger than 100, and false(0) otherwise.

```
(data <0  || data > 100)
```

b) Write a Boolean expression that is true(1) when the value of `data` is between 3 and 5 inclusive (i.e., including 3 and 5), and false(0) otherwise.

```
(data >= 3  && data <= 5)
```

c) Write a Boolean expression that is true(1) when the value of `data` is equal to 0 or 1, and false(0) otherwise.

```
(data == 0  || data == 1)
```

4. Consider the following code segment:

```
if( a > 0 && b > 0 )
    a = a + 2;

if( 2 * a + b > 20 )
    a = a + 4;
else {
    a = a + 1;
    b = b + 1;
}
```

What are the values of `a` and `b` after this code has executed assuming that we start with the values…

i) 2 for `a` and 4 for `b`

```
a will be 5
b will be 5
```

ii) 8 for `a` and 5 for `b`

```
a will be 14
b will be 5
```

iii) 20 for `a` and -2 for `b`

```
a will be 24
b will be -2
```

iv) -2 for `a` and 5 for `b`

```
a will be -1
b will be 6
```

5. Consider the following code segment:

```
if( a > 0 && b > 0 ) {
    a = a + 2;

    if( 2 * a + b > 20 )
        a = a + 4;
}
else {
    a = a + 1;
    b = b + 1;
}
```

What are the values of `a` and `b` after this code has executed assuming that we start with the values…

i) 2 for `a` and 4 for `b`

```
a will be 4
b will be 4
```

ii) 8 for `a` and 5 for `b`

```
a will be 14
b will be 5
```

iii) 20 for `a` and -2 for `b`

```
a will be 21
b will be -1
```

iv) -2 for `a` and 5 for `b`

```
a will be -1
b will be 6
```

6. Suppose that `data` is a variable of type `int`. Further suppose that you've been asked to write a Boolean expression that is true if the value of `data` is neither 0 nor 1, and false otherwise. Chris and Pat have come up with the following answers:

Chris: `!( data == 0 || data == 1 )`

Pat: `( data != 0 && data != 1 )`

Who is right?

If these answers are correct, when `data` has the value 0 or 1, the expression must produce false. For any other value of `data`, it must produce true.

Chris

Data	`( data == 0 )`	`( data == 1 )`	`!(( data == 0 ) || ( data == 1 ))`	Outcome
0	T	F	T	F
1	F	T	T	F
2	F	F	F	T

Pat

Data	`data != 0`	`data != 1`	`( data != 0 && data != 1 )`	Outcome
0	F	T	F	F
1	T	F	F	F
2	T	T	T	T

So both Chris and Pat have produced an answer that is correct. Note that this is a case of DeMorgan's Law. In general, DeMorgan's Law states that if `b1` and `b2` are Boolean expressions,

`!( b1 || b2 )` is equivalent to `( !b1 && !b2 )` and

`!( b1 && b2 )` is equivalent to `( !b1 || !b2 )`

In this case `b1` is the expression `( data == 0 )` and `b2` is the expression `( data == 1 )`.

Chris: `!( ( data == 0 ) || ( data == 1 ) )`

Pat: `( data != 0 && data != 1 )`

One thing that you do need to watch out for is that `data` is of type int. So it can be any of the integer values in the red area.

7. Complete the function below so that it computes income tax based the given `income` argument. Be sure to make use of the symbolic constants AND ensure you do not introduce redundant conditions.

Income	Tax payable
income $\leq$ 12500	0%
12500 $<$ income $\leq$ 65000	24% of income in excess of $12,500
income $>$ 65000	$12,600 plus 32% of income in excess of $65,000

```
/*
 * Author: Instructor
 * Purpose: provides income cutoffs, tax rates and function to calculate of tax owing
 */

#include <stdio.h>

#define BASE_CUTOFF   12500.0
#define MID_CUTOFF    65000.0
#define MID_RATE      0.24
#define HIGH_RATE     0.32
#define HIGH_BASE_TAX 12600.0

void print_tax_owing(double income);

int main( void ) {
    // calls to test the function under, at and above all boundaries:
    print_tax_owing(BASE_CUTOFF - 1); // should print Tax owing: $0.00
    print_tax_owing(BASE_CUTOFF); // should print Tax owing: $0.00
    print_tax_owing(BASE_CUTOFF + 1); // should print Tax owing: $0.24
    print_tax_owing(MID_CUTOFF - 1); // should print Tax owing: $12599.76
    print_tax_owing(MID_CUTOFF); // should print Tax owing: $12600.00
    print_tax_owing(MID_CUTOFF + 1); // should print Tax owing: $12600.32

    return 0;
}

/*
 * Purpose: calculate the tax owed on given income
 * Parameters: double income – yearly income in Canadian dollars
    */
void print_tax_owing(double income){
    double tax_owing;

    if ( income <= BASE_CUTOFF ){
           tax_owing = 0.0;
    } else if ( income <= MID_CUTOFF ) {
        tax_owing = ( income - BASE_CUTOFF ) * MID_RATE;
    } else {
        tax_owing = HIGH_BASE_TAX +
                    ( income - MID_CUTOFF ) * HIGH_RATE;
    }

    printf( "Tax owing: $%.2f\n", tax_owing );
}
```

8. Design a function that takes air temperature (in degrees Celsius) and pressure measurements (in pounds per square inch, psi) of a machine and indicates the operating conditions. The function should print the message "Error: data not valid" if the pressure is negative, otherwise it prints a message about the machine's operation according to the following:
   - If the temperature is above 300^{o}C or below 5^{o}C, or if the pressure is above 150psi, the machine is not operating under normal conditions.
   - If the temperature is above 250^{o}C and the pressure is above 100psi, the machine is not operating under normal conditions.
   - the machine is operating under normal conditions.

   Be sure to include:

   - appropriate documentation / comment statements
   - the definition of symbolic constants, where appropriate
   - test your code to ensure it is correct

```c
#include <stdio.h>

#define MAX_TEMP            300.0
#define MAX_PRESSURE        150.0
#define TEMP_THRESHOLD      250.0
#define PRESSURE_THRESHOLD  100.0
#define MIN_TEMP            5.0
#define MIN_PRESSURE        0.0

void check_reading(double temp, double pressure);

int main( void ) {
    // call function to test - ensure you test all combinations of the conditions
    printf("\nshould all print Error: Data not valid\n");
    check_reading(MIN_TEMP-1, -1);
    check_reading(MIN_TEMP, -1);
    check_reading(MAX_TEMP, -1);
    check_reading(MAX_TEMP+1, -1);

    printf("\nshould print Machine not operating under normal conditions\n");
    check_reading(MIN_TEMP-1, 0);
    check_reading(MAX_TEMP+1, 0);
    check_reading(MIN_TEMP-1, MAX_PRESSURE);
    check_reading(MAX_TEMP+1, MAX_PRESSURE);
    check_reading(MIN_TEMP-1, MAX_PRESSURE+1);
    check_reading(MIN_TEMP, MAX_PRESSURE+1);
    check_reading(MAX_TEMP, MAX_PRESSURE+1);
    check_reading(MAX_TEMP+1, MAX_PRESSURE+1);

    printf("\nshould print Machine not operating under normal conditions\n");
    check_reading(TEMP_THRESHOLD+1, PRESSURE_THRESHOLD+1);

    printf("\nshould print Machine operating under normal conditions\n");
    check_reading(MIN_TEMP, MAX_PRESSURE-1);
    check_reading(MIN_TEMP+1, MAX_PRESSURE-1);
    check_reading(MIN_TEMP, MAX_PRESSURE);
    check_reading(MIN_TEMP+1, MAX_PRESSURE);
    check_reading(TEMP_THRESHOLD-1, PRESSURE_THRESHOLD);
    check_reading(TEMP_THRESHOLD, PRESSURE_THRESHOLD);
    check_reading(TEMP_THRESHOLD+1, PRESSURE_THRESHOLD);
    check_reading(TEMP_THRESHOLD-1, PRESSURE_THRESHOLD-1);
    check_reading(TEMP_THRESHOLD, PRESSURE_THRESHOLD-1);
    check_reading(TEMP_THRESHOLD+1, PRESSURE_THRESHOLD-1);

    return 0;
}

/*
 * Purpose: determine if temp and pressure readings are acceptable
 * Parameters: double temp - machine temperature in Celsius
 *             double pressure - machine pressure in lbs/sq inch
 */
void check_reading (double temp, double pressure) {
    if ( pressure < 0 ) {
        printf("Error: data not valid\n");
    } else {
        if( temp > MAX_TEMP || temp < MIN_TEMP || pressure > MAX_PRESSURE ) {
            printf("Machine is NOT operating under normal conditions\n");
        } else if( temp > TEMP_THRESHOLD  &&  pressure > PRESSURE_THRESHOLD ) {
            printf("Machine is NOT operating under normal conditions\n");
        } else {
            printf("Machine is operating under normal conditions\n");
        }
    }
}
```