A+ Computer Science

BOOLEAN LOGIC

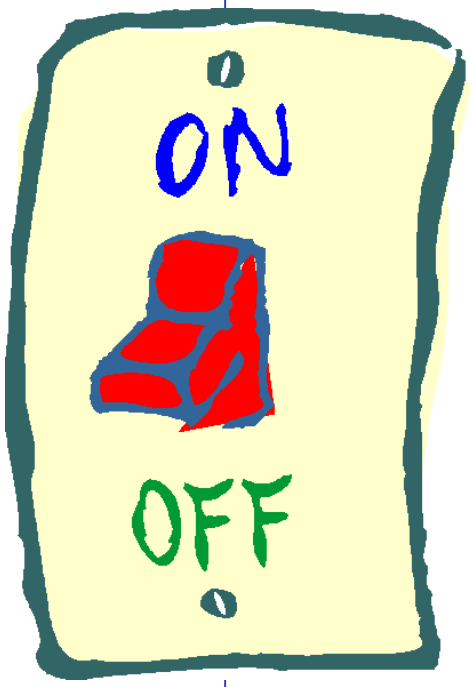
What is a boolean?

A boolean is any condition or variable that can be evaluated to true or false.

```
boolean stop = false;
boolean go = true;

if(x>10) {  }

while(z<20) { }
```



Precedence

()	HIGH
! ++ --	
* / %	
+ -	
<< >> (bitwise shifts)	
< <= > >=	
== !=	
& (bitwise and)	
^ (bitwise xor)	
| (bitwise or)	
&& (logical and)	
|| (logical or)	
= += -= *= /= %=	
,	LOW

Boolean Symbols

Name	Boolean Symbol	Java Counterpart
and	∧ logical and	&&
or	∨ logical or	||
not	¬ logical not	!

Logical AND &&

&&

all conditions must be true

```
if (total==17 && 92==num)
{
  do something 1;
  do something 2;
}
```

Logical AND &&

&&

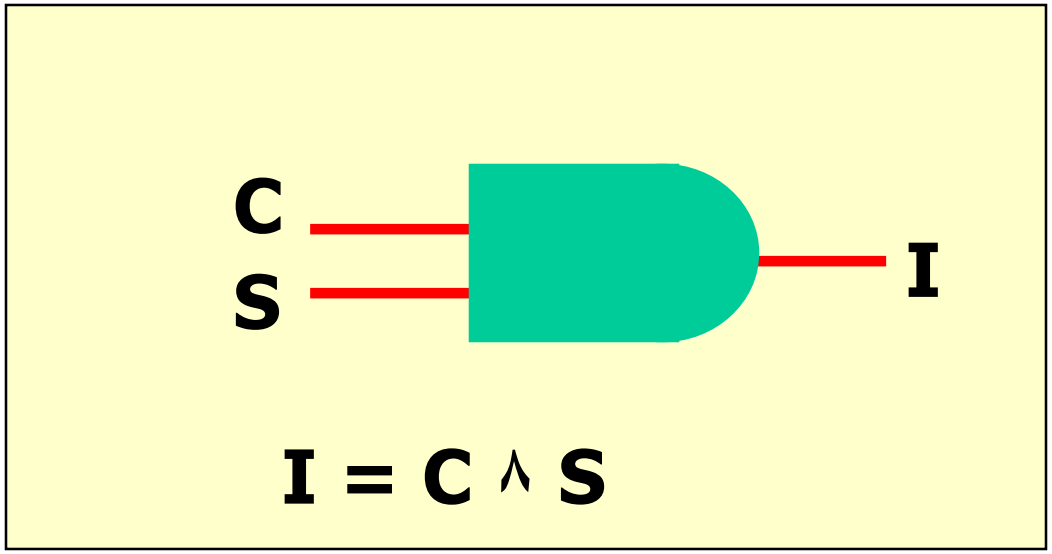
all conditions must be true

```
if ( C == true
        &&   S == true )
{
  do something 1;
  do something 2;
}
```

C	S	I
0	0	0
0	1	0
1	0	0
1	1	1

Logical AND &&

Engineering Symbol



C	S	I
0	0	0
0	1	0
1	0	0
1	1	1

Logical OR | |

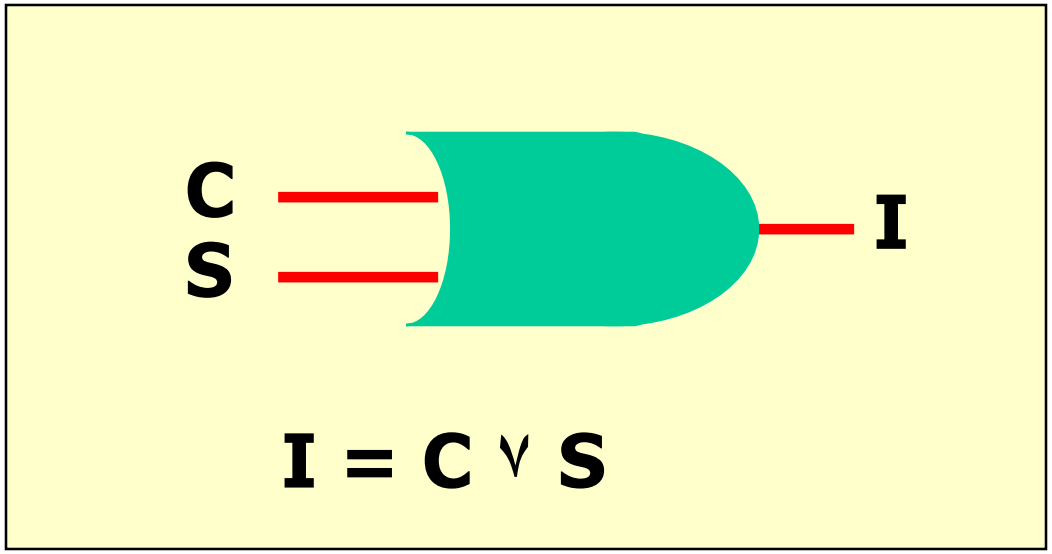
| |

any condition can be true

```
if (total==9 || num==31)
{
   do something 1;
   do something 2;
}
```

Logical OR | |

Engineering Symbol



C	S	I
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Rules

true and false = false
false and true = false
false and false = false
true and true = true

false or true = true
true or false = true
true or true = true
false or false = false

Logical NOT !

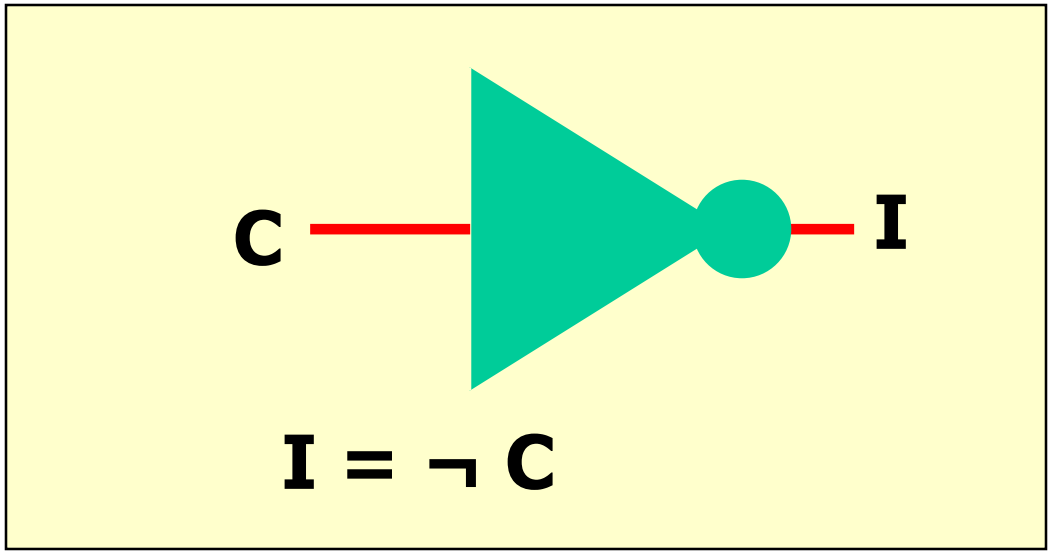
!

true (if condition is false)

```
if (! pass.equals("pass"))
{
   do something 1;
   do something 2;
}
```

Logical NOT !

Engineering Symbol



C	I
0	1
1	0

Logical XOR ^

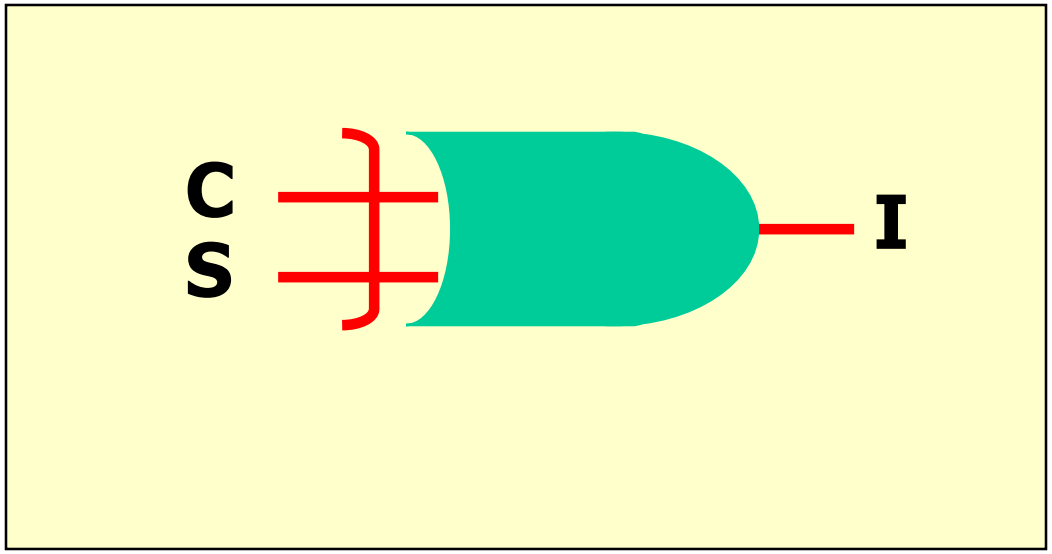
^

true if only one condition is true

```
if (total==34 ^ num==23)
{
   do something 1;
   do something 2;
}
```

Logical XOR ^

Engineering Symbol



C	S	I
0	0	0
0	1	1
1	0	1
1	1	0

logical.java

Nesting Ifs

```
int num=7;
if(num>2)
{
   if(num<10)
      System.out.println(">2<10");
   if(num>10)
      System.out.println(">2>10");
}
```

OUTPUT

```
>2<10
```

Nesting Ifs

```
int num=17;
if(num>2)
{
   if(num<10)
      System.out.println(">2<10");
   else
      System.out.println("fun");
}
```

OUTPUT

fun

Nesting Ifs

```
int x = 10;
if( x >= 3 ){
  System.out.println("aplus");
  if( x >= 10 )
    System.out.println("comp");
  if( x < 10 )
    System.out.println("sci");
}
System.out.println("rox");
```

OUTPUT

```
aplus
comp
rox
```

Nesting Ifs

```
int x = -5;
if( x >= 3 ){
  System.out.println("aplus");
  if( x >= 10 )
    System.out.println("comp");
  if( x < 10 )
    System.out.println("sci");
}
System.out.println("rox");
```

OUTPUT

```
rox
```

ifnesting.java

Common Boolean Laws

DeMorgan's Law

$\neg(C \vee S) = \neg C \wedge \neg S$ DeMorgan's Law

$\neg(C \wedge S) = \neg C \vee \neg S$ DeMorgan's Law

```
!(c||s) == !c&&!s    Java Code
!(c&&s) == !c||!s    Java Code
```

This is always used by AP and UIL!

DeMorgan's Law

```
boolean C = true;
boolean S = true;
boolean I = !( C && S );
System.out.println( I );
```

C	S	I
1	1	0
1	0	1
0	1	1
0	0	1

OUTPUT

false

DeMorgan's Law

```
boolean C = false;
boolean S = true;
boolean I = !( C || S);
System.out.println( I );
```

C	S	I
1	1	0
1	0	0
0	1	0
0	0	1

OUTPUT

false

Truth Table Example

Which statement is represented by the truth table at right?

A. `I = !(C&&S)&&(C||S);`
B. `I = C||S&&S;`
C. `I = C&&S;`
D. `I != C&&S;`

C	S	I
0	0	0
0	1	1
1	0	1
1	1	1

demorganslaw.java

Work on Programs!

Crank Some Code!

More Boolean Laws

Law of Absorption

$C \wedge (C \vee S) = C$	Law of Absorption
$C \vee (C \wedge S) = C$	Law of Absorption

c&&(c||s)	Java Code
c||(c&&s)	Java Code

This is used now and again by AP and UIL!

Truth Table Example

```
boolean C = true;
boolean S = false;
boolean I = C || ( C && S );
System.out.println( I );
```

C	S	I
1	1	1
1	0	1
0	1	0
0	0	0

OUTPUT

true

Truth Table Example

```
boolean C = false;
boolean S = true;
boolean  I = C && ( C || S );
System.out.println( I );
```

C	S	I
1	1	1
1	0	1
0	1	0
0	0	0

OUTPUT

false

absorptionlaw.java

Distributive Law

$C \wedge (S \vee I) = (C \wedge S) \vee (C \wedge I)$ Distributive

$C \vee (S \wedge I) = (C \vee S) \wedge (C \vee I)$ Distributive

```
c&&(s||i)
is the same as
(c&&s)||(c&&i)
```

This is used now and again by AP and UIL!

Distributive Law

```
boolean C=true, S=true, I=false, ans;

ans=(( C || ( S && I ))
          ==
       (( C || S ) && ( C || I )));

System.out.println(ans);
```

OUTPUT

true

distributivelaw.java

Short Circuit Evaluation

Short Circuit Evaluation

Java evaluates boolean expressions from left to right in most situations and stops the evaluation process once a condition is found that can complete the expression.

&& - and
|| - or

Short Circuit Evaluation

```
int total=9;
boolean flipper = false;

if(flipper || total>4)
{
  out.println("short");
}
out.println("check");
```

OUTPUT

```
short
check
```

Short Circuit Evaluation

```
int total=2;
boolean flipper = true;

if(flipper || total>4)
{
  out.println("short");
}
out.println("check");
```

OUTPUT

```
short
check
```

Short Circuit Evaluation

```
int total=2;
boolean flipper = false;

if(flipper || total>4)
{
  out.println("short");
}
out.println("check");
```

OUTPUT

```
check
```

Short Circuit Evaluation

```
int total=9, num=13;

if (total<4 || ++num<15)
{
    out.println("short");
}
out.println(num);
```

OUTPUT

```
short
14
```

Short Circuit Evaluation

```
int total=9, num=13;

if (total>4 && ++num>15)
{
    out.println("short");
}
out.println(num);
```

OUTPUT

14

Short Circuit Evaluation

```
int total=9, num=13;

if (total>4 ||++num>15 && total>0)
{
    out.println("short");
}
out.println(num);
```

OUTPUT

```
short
13
```

The && never happens!

shortone.java
shorttwo.java
shortthree.java
shortfour.java

Random Numbers

Random Numbers

```
double decOne;
decOne = Math.random() * 10;
int intOne;
intOne = (int)(Math.random() * 10);

System.out.println(decOne);
System.out.println(intOne);
```

OUTPUT

```
8.441931676606826
```

Random Numbers

reference variable

```
Random rand = new Random();
```

object instantiation

Always make Random vars instance vars!

Random

frequently used methods

Name	Use
nextInt(x)	returns a random int 0 to x(exclusive)
nextInt()	returns a random int MIN to MAX(exclusive)
nextDouble()	returns a random int 0.0 to 1.0(exclusive)

```
import  java.util.Random;
```

Random Numbers

```
Random rand = new Random();
int intOne = rand.nextInt(10);          //0-9
System.out.println(intOne);
intOne = rand.nextInt(50)+1;            //1-50
System.out.println(intOne);
intOne = rand.nextInt(20)+20;           //20-39
System.out.println(intOne);
```

OUTPUT

```
7
29
37
```

randomone.java

dowhile.java
password.java

George Boole

George Boole's work is considered by many the starting point of Boolean Algebra. His work is also considered as a beginning of sorts for Comp Sci.

Alice in Wonderland

Lewis Carroll

Work on Programs!

Crank Some Code!

A+ Computer Science

BOOLEAN LOGIC