# Tutorial 3- Solution

**Objectives:** To practice with

* Relational and logical operators
* if…else and switch selection statements

## Suppose integer variables x=3, y=0, z=-4, what is the value of each of the following expressions true or false?

|  |  |  |  |
| --- | --- | --- | --- |
| a. x >= 0 && y <= b. x != y || x != c. ++x > 3 && y++ | 0  z  == 0 | Answer: Answer: Answer: | **true true true** |
| d. !(x != y) |  | Answer: | ***false*** |
| *e.* x > 0 && 'B' < | 'A' | Answer: | ***false*** |

**Further clarification:**

Using the C language operators’ precedence and associativity, the above expressions are evaluated as follows

|  |
| --- |
| 1. x >= 0 && y <= b as (x >= 0) && (y <= b) 2. x != y || x != z as (x != y) || (x != z) 3. ++x > 3 && y++ ==0 as (++x > 3) && (y++ ==0), the old current value of y is used first in the expression before being incremented at the end   d. !(x != y)  *e.* x > 0 && 'B' <'A' as (x > 0) && ('B' <'A') |

|  |
| --- |
| **Operators Precedence** |
| function calls, (…), postfix ++, -- |
| ! + - & (unary operator) |
| \* / % |
| + - |
| < <= >= > |
| == != |
| && |
| || |
| = |

1. **What is Short-Circuit Evaluation? Evaluate the following expressions and variables?**

bool flag; int x, y;

a. x=y=10;

flag = x>0 || y++;

**y = 10**

**y = 10**

**y = 11**

**flag = true**

**flag = false**

**flag = true**

b. x=y=10;

flag = x<0 && --y;

c. x=y=10;

if( x==10 && ++y>10 ) flag = true;

else flag= false;

Answer:

Please refer to the precedence operators table above, or check the most complete one in Lecture 1, slide 46 or condstat\_supp , slide 7

Short-Circuit Evaluation: C stops further evaluation of an expression when the result becomes obvious.

* x=y=10 is equivalent to x=(y=10), associativity of ‘=’ is from right to left, which leads x=10;

The expression x>0 || y++; is grouped as (x>0) || (y++);

in the *flag* statement, x>0 is true and given the || operator, the right side of the expression won’t be evaluated as the while expression is true regardless. Therefore, y is not updated.

* In this case, the expression x<0 && --y; is grouped as (x<0) && (--y);

Given the value of x, (x<0) is false, therefore the whole expression is false and --y” is

not executed.

* The expression x==10 && ++y>10 corresponds to (x==10) && ((++yy)>10)

As X==10 is true , the (++yy>10) will be evaluated to calculate the Boolean value of the whole and expression. Given the precedence of the operator, y gets incremented first before being compared to 10. This sets the whole expression to True.

## Suppose an integer number = 5 What is the output:

**a.** if(number < 3)

printf("1\n"); else if(number == 5)

printf("2\n");

else

printf("3\n");

**2**

Answer:

**b.** if(number < 3)

if(number == 5)

printf("1\n");

else

printf("2\n");

else

printf("3\n");

**3**

Answer:

**.** if(number < 3)

if(number == 5)

printf("1\n");

else

printf("2\n");

else

printf("3\n");

would give the same output. The compiler going through the lines of the program as per their order, pairs “else” with the first free preceding “if”

## Suppose integers x=0, y=0, z=1. What are the values of x, y and z after executing the code:

switch ( x )

{

case 0: y=2;

z=3;

case 1: y=4;

break; default: z=0;

}

**x = 0 y = 4 z = 3**

Answer:

Case 0 statements are executed, since we did not add a “break; statement”, the following statements in the switch case will get executed unless a” break; statement” is encountered; it is the break; statement which takes the code out of the switch block

1. **Write an expression to test for each of the following relationships.**
2. Age is from 18 to 21 inclusive.
3. water is less than 1.5 and also greater than 0.1.
4. year is divisible by 4. (*Hint:* Use %.)
5. speed is not greater than 55.
6. y is greater than x and less than z.
7. w is either equal to 6 or not greater than 3.
8. age >= 18 && age <= 21

The expression 18<=age<=21 would be equivalent to (18<=age)<=21; its resulting value is therefore always true as 18<= age can be either 0 or 1 which is always <=21

1. water < 1.5 && water > 0.1
2. year % 4 == 0
3. speed <= 55
4. y > x && y < z
5. w == 6 || w <= 3
6. **Write assignment statements for the following:**
7. Assign a value of 0 to between if n is less than −k or greater than +k; otherwise, assign 1.
8. Assign a value of 1 to divisor if digit is a divisor of num; otherwise, assign a value of 0, including the case digit = 0.
9. Assign a value of 1 to lowercase if ch is a lowercase letter; otherwise, a value of 0.
10. if (n < -k || n > k)

between = 0;

else

between = 1;

The value of between is hence the opposite of the if condition, we can thus write: between = !(n < -k || n > k);

1. if (digit == 0) /\* check for divide by 0 before attempting division \*/

divisor = 0;

else if (num % digit == 0)

divisor = 1;

else

divisor = 0;

another elegant answer:

switch(digit){

case 0: divisor =0; break;

default: if (num % digit == 0)

divisor = 1;

else

divisor = 0;

}

1. if (ch >= 'a' && ch <= 'z')

lowercase = 1;

else

lowercase = 0;

1. **Write an if statement that displays an acceptance message for an astronaut candidate if the person’s weight is between the values of opt\_min and opt\_max inclusive, the person’s age is between age\_min and age\_max inclusive, and the person is a nonsmoker (smoker is false).**

if (weight >= opt\_min && weight <= opt\_max && age >= age\_min &&

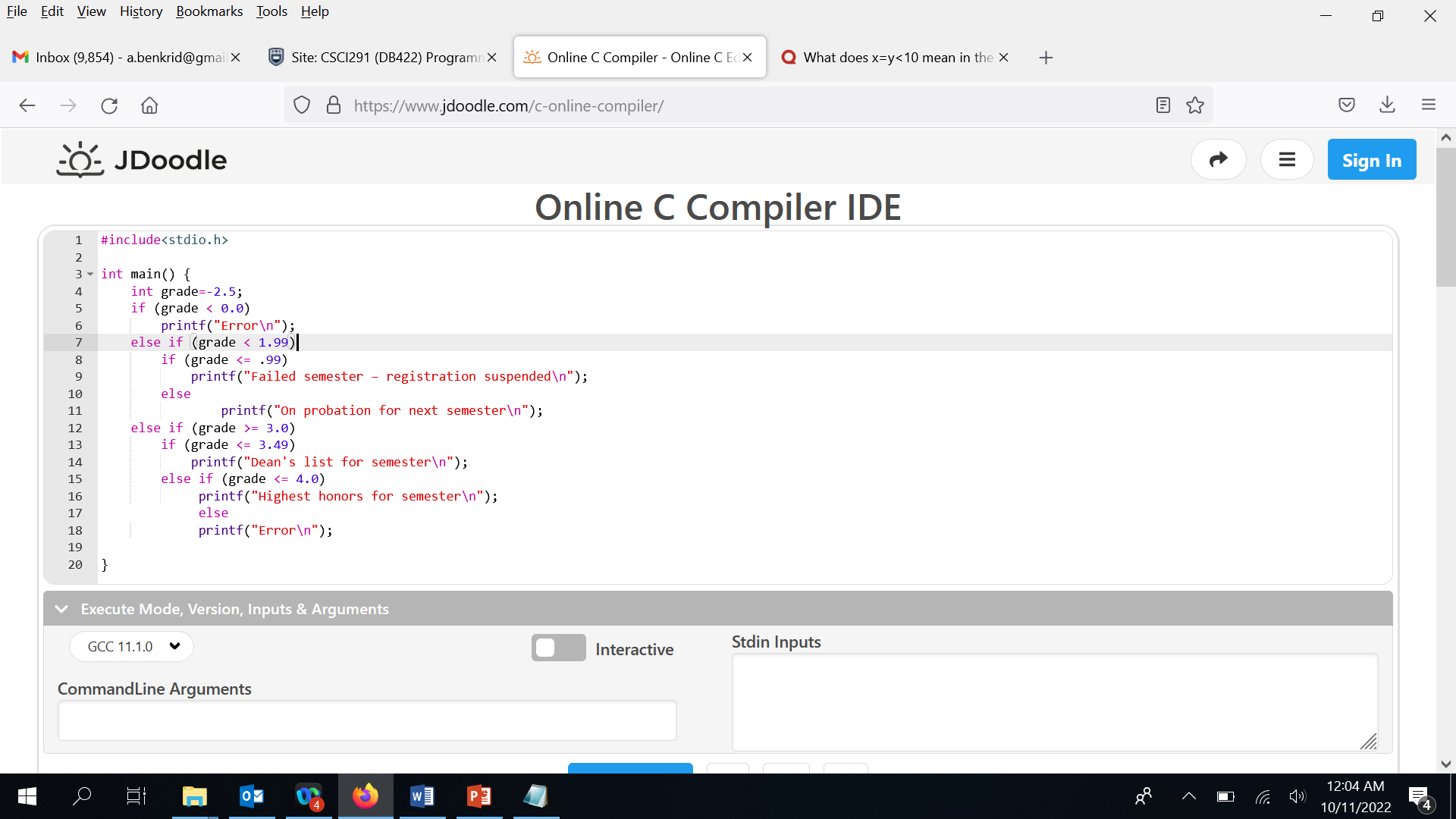
age <= age\_max && !smoker)

printf("Acceptable astronaut candidate.\n");

1. **Implement the following decision table using a nested if statement. Assume that the grade point average is within the range 0.0 through 4.0.**

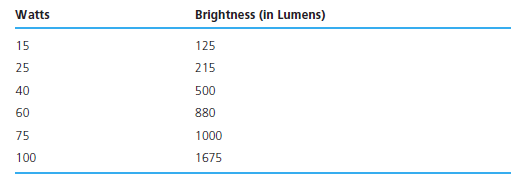


Answer:



Good indentation helps a lot to understand the correct “else” - “if” pairing

1. **Write a switch statement that assigns to the variable lumens the expected brightness of a standard light bulb whose wattage has been stored in watts. Use this table:**



**Assign −1 to lumens if the value of watts is not in the table.**

switch (watts) {

case 15:

lumens = 125;

break;

case 25:

lumens = 215;

break;

case 40:

lumens = 500;

break;

case 60:

lumens = 880;

break;

case 75:

lumens = 1000;

break;

case 100:

lumens = 1675;

break;

default:

printf("\nError: Unknown lumens.");

lumens = -1;}

1. **Implement the flow diagram in below Fig. using a nested if structure.**



if (age > 59)

if ( sts == 'W')

printf("Working senior\n");

else

printf("Retired senior\n");

else if (age > 20)

printf("Adult\n");

else if (age > 12)

printf("Teen\n");

else

printf("Child\n");

1. **Write an interactive program that contains an if statement that may be used to compute the area of a square (area= *side2*) or a circle (area= π × *radius2*) after prompting the user to type the first character of the figure name (S or C).**

#include <stdio.h>

/\* Different type shapes to compute area for. \*/

#define SELECT\_CIRCLE 'C' /\* Type Circle \*/

#define SELECT\_SQUARE 'S' /\* Type Square \*/

#define PI 3.14159265

int main(void)

{

char type; /\* Type of shape. \*/

double area, side\_base, radius;

/\* Display menu and get the type of shape to compute the area. \*/

printf("Enter a:\n");

printf(" %c -- To compute the area of a circle.\n",

SELECT\_CIRCLE);

printf(" %c -- To compute the area of a square.\n",

SELECT\_SQUARE);

printf("Select> ");

scanf("%c", &type);

/\* Compute the area of different shapes. \*/

if (type == SELECT\_CIRCLE) {

/\* Get information for CIRCLE radius needed to compute the area. \*/

printf("Enter radius> ");

scanf("%lf", &radius);

area = PI \* radius \* radius;

printf("The area of the circle is %.4f\n", area);

} else {

/\* Get information for SQUARE edge needed to

compute the area. \*/

if (type == SELECT\_SQUARE) {

printf("Enter side> ");

scanf("%d", &side\_base);

area = side\_base \* side\_base;

printf("The area of the square is %d\n", area);

} else {

printf("ERROR: Invalid selection.\n");

}

}

return (0);

}

**See attached tut3\_ex11 for another solution to this question.**