Ocean Lu  
CS 4080.02  
Doctor Yang  
09.26.2019

ICE 3

**Problem 1:**

* For each of the following Java-like Boolean expressions,
  1. indicate the evaluation order under strict evaluation.
  2. indicate the evaluation order under short-circuit evaluation.
  3. what is the result of the expression if evaluated by Java?

1. a=5, b=3, c=30, tag = true;   
   (a < b) || !tag && ( b / (2\*a\*b-c)> 0)
2. a<b = 5 < 3 = false  
   !tag = false  
   b/(2\*a\*b-c)>0 = can’t divide by zero  
   = false
3. a<b = 5 < 3 = false  
   = false
4. False
5. a=5, b=3, c=30, tag = false;   
   (a > b) || !tag && ( b / (2\*a\*b-c)>0)
6. a>b = 5 > 3 = true

!tag = true

b/(2\*a\*b-c)>0 = can’t divide by zero

= true

1. a>b = 5 > 3 = true  
   = true
2. True

**Problem 2:**

* What is the difference between == and === in JavaScript?  
  The ‘==’ operator tests for abstract equality, while the ‘===’ operator tests for strict equality.
* Name an advantage of using assignment as an expression, then give an example (a line of code) to support it.  
  An advantage of using assignment as an expression is that
* What will be printed by a Python shell?  
  >>a, b = 1,4  
  >>b,a = a+b,b-a  
  >>print (a,b)  
  The following is printed as: 3 5. This is because a, assigned as 1, becomes (b-a), which is 4-1 that becomes 3. For b, assigned as 4, becomes (a+b), which is 1+5 that becomes 5.

**Problem 3:**

Use Java or C++ to rewrite the following pseudo-code segment using a loop structure without goto, break, or any other unconditional branching statement:  
k = (j+13)/27; //assume i,j,k are integers properly declared.  
loop:  
 if k > 10 then goto out  
 k=k+1.2;  
 i=3\*k-1;  
 goto loop;  
out: …  
k = (j+13)/27;  
while (k>10) {  
 k = k + 1;  
 i = 3 \* k – 1;  
}

**Problem 4:**

* Use C++ or Java to rewrite the following code segment using a switch statement. Do not optimize the code, just do a direct translation to switch statement.  
  if ((k == 1) || (k == 2)) j = 2\*k -1  
  if ((k==3) || (k==5)) j=3\*k+1  
  if (k==4) j=4\*k-1  
  if ((k==6) || (k==7) || (k==8)) j=k-2

switch(k) {  
 case 1:   
 case 2: j = 2 \* k – 1;   
 break;  
 case 3:   
 case 5: j = 3 \* k + 1;   
 break;  
 case 4: j = 4 \* k – 1;   
 break;  
 case 6:   
 case 7:   
 case 8: j = k + 2;  
}

Problem 5:

Use Java or C++ to rewrite the following code without using goto, break or any other unconditional branching statement. Make sure the revised code with the same complexity as the given code (i.e. the # of comparisons performed in the if statement should be almost same.)   
for (i=1; i<=n; i++) {  
 for (j=1; j<=n; j++)  
 if (x[i][j] != 0) goto reject;  
 println(“First all –zero row is:” i);  
 break;  
reject:  
}

boolean false = false;  
for (i = 1; i <= n; i++) {  
 int counter = 0;   
 for (j = 1; j <= n; j++) {  
 if (x[i][j] == 0)   
 counter++;  
 }  
 if (counter == n && found == false) {  
 prinf(“First all-zero row is: %d”, i);   
 found = true;  
 }  
}