

Chapter 3 Test Review

Monday, November 05, 2012

7:55 AM

AP Computer Science

Chapter 3 Test Review

Name: _____

The next two questions refer to this code segment

```
if (a != b && n / (a - b) > 90)
{
    /* statement 1 */
}
else
{
    /* statement 2 */
}
```

Free Response - Be prepared for:

1. Write a sentinel value, keyboard loop
2. Write a count for loop
3. Write a most-wanted-holder pattern
4. Write an accumulator pattern

1. What will happen if $a == b$ is true?

- a) /* statement 2 */ will be executed
b) /* statement 1 */ will be executed
c) Neither /* statement 1 */ nor /* statement 2 */ will be executed.
d) A compile-time error will occur.
e) An exception will be thrown.

2. What will happen if $a == b$ is false?

- a) /* statement 1 */ will be executed
b) /* statement 2 */ will be executed
c) Either /* statement 1 */ or /* statement 2 */ will be executed.
d) A compile-time error will occur.
e) An exception will be thrown.

3. Look at the following poorly formatted program segment. If $a = 7$ and $c = 6$ before execution, which of the following represents the correct values of c , d , p , and t after execution? An undetermined value is represented with a question mark.

f

```
if (a == 6)
{
    if (c == 6)
    {
        c = 9;
        d = 9;
    }
    else
    {
        t = 10;
        if (c == 6)
            c = 5;
    }
}
else p = 9;
```

$\frac{9}{9}$ $\frac{c}{c}$ $\frac{d}{9}$ $\frac{p}{9}$ $\frac{t}{9}$

- a) $c = 6$, $d = ?$, $p = 9$, $t = ?$
b) $c = 5$, $d = ?$, $p = ?$, $t = 10$
c) $c = 6$, $d = ?$, $p = ?$, $t = ?$
d) $c = 5$, $d = 9$, $p = ?$, $t = 10$
e) $c = 9$, $d = 9$, $p = ?$, $t = ?$

4. Consider the following outline of a nested if-else structure which has more if clauses than else clauses. Which of the statements below is true regarding this structure?

```
if (condition1)
  if (condition2)
    statement1;
  else statement2;
```

- a) syntactically it is invalid to have more if clauses than else clauses
b) statement2 will only execute if condition1 is false and condition2 is false
☒ c) statement2 will only execute if condition1 is true and condition2 is false
d) statement2 will only execute if condition1 is false, it does not matter what condition2 is
e) statement2 will never execute
5. What values are stored in x and y after execution of the following program segment?

```
int x = 30, y = 40;
if (x >= 0)
{
  if (x <= 100)
  {
    y = x * 3;
    if (y < 50)
      x /= 10;
  }
  else
    y = x * 2;
}
else
  y = x;
```

Handwritten notes: $\frac{x}{30}$, $\frac{y}{40}$, $\frac{y}{90}$, y sets set to 90, $x \neq 10$, $y = x$.

- ☒ a) x = 30 y = 90
b) x = 30 y = -30
c) x = 30 y = 60
d) x = 3 y = -3
e) x = 30 y = 40

6. Of the following if statements, which one correctly executes three instructions if the condition is true, and executes no instructions if it is false?

~~a)~~ `if (x < 0)`
`a = b * 2;`
`y = x;`
`z = a - y;`

~~b)~~ `if (x < 0)`
`a = b * 2;`
`y = x;`
`z = a - y;`
`}`

~~c)~~ `if ((x < 0))`
`a = b * 2;`
`y = x;`
`z = a - y;`
`}`

d) `if (x < 0) {`
`a = b * 2;`
`y = x;`
`z = a - y;`
`}`

e) b, c and d are all correct, but not a

7. Consider the following code that will assign a letter grade of A, B, C, D, or F depending on a student's test score.

```
if (score >= 90)
    grade = "A";
if (score >= 80)
    grade = "B";
if (score >= 70)
    grade = "C";
if (score >= 60)
    grade = "D";
else
    grade = 'F';
```

Handwritten analysis of the code:

92	→	A B D
82	→	D
72	→	D
62	→	D
52	→	F

- a) This code will work correctly in all cases
 b) This code will work correctly only if `grade >= 60`
 c) This code will work correctly only if `grade < 60`
d) This code will work correctly only if `grade < 70`
 e) This code will not work correctly under any circumstances

8. What is wrong, logically, with the following code?

```
if (x > 10)
    System.out.println("Large");
else if (x > 6 && x <= 10)
    System.out.println("Medium");
else if (x > 3 && x <= 6)
    System.out.println("Small");
else
    System.out.println("Very small");
```

→ Redundant

- a) There is no logic error, but there is no need to have $(x \leq 10)$ in the second conditional or $(x \leq 6)$ in the third conditional
- b) There is no logic error, but there is no need to have $(x > 6)$ in the second conditional or $(x > 3)$ in the third conditional
- c) The logic error is that no matter what value x is, "Very small" is always printed out
- d) The logic error is that no matter what value x is, "Large" is always printed out
- e) There is nothing wrong with the logic at all

9. Given that a , b , and c are integers, consider the boolean expression

$(a < b) \ || \ || \ ((c == a * b) \ \&\& \ (c < a))$

? || ! (+ + +) -

Which of the following will *guarantee* that the expression is true?

- a) $c < a$ is false. **T**
- b) $c < a$ is true. we don't know
- c) $a < b$ is false. ?
- d) $c == a * b$ is true. ?
- e) $c == a * b$ is true, and $c < a$ is true. ?

? || ! A

? || f.

10. Which of the following will evaluate to true only if boolean expressions A , B , and C are all false?

- a) $!A \ \&\& \ !B \ \&\& \ !C$ → **T**, there are other values A, B, C that would make this true
- b) $!A \ || \ !B \ || \ !C$ → T, "
- c) $!(A \ || \ B \ || \ C)$ → T, and $A/B/C$ all false is the only way this can be true
- d) $!(A \ \&\& \ B \ \&\& \ C)$ → T, but there are other ways
- e) $!A \ || \ !(B \ || \ !C)$ → T, but there are other ways...

11. Assume that `x` and `y` are `int` variables with `x = 5`, `y = 3`, and `a` and `d` are `String` variables with `a = "a"` and `d = "A"`, and examine the following conditions:

Condition 1: `(x < y && x > 0)`

✓ Condition 2: `(!a.equals(d) || x != 5)`

✓ Condition 3: `!(true && false)`

Condition 4: `(x > y || a.equals("A") || !d.equals("A"))`

- a) All 4 Conditions are true
 - b) Only Condition 2 is true
 - c) Condition 2 and Condition 4 are true only
 - ✓ d) Conditions 2, 3 and 4 are all true, Condition 1 is not
 - e) All 4 Conditions are false
12. Assume that `x` and `y` have been declared and initialized with `int` values. Consider the following Java expression

```
(y > 10000) || (x > 1000 && x < 1500)
```

Which of the following is equivalent to the expression given above?

- ✓ a) `(y > 10000 || x > 1000) && (y > 10000 || x < 1500)`
- b) `(y > 10000 || x > 1000) || (y > 10000 || x < 1500)`
- c) `(y > 10000) && (x > 1000 || x < 1500)`
- d) `(y > 10000 && x > 1000) || (y > 10000 && x < 1500)`
- e) `(y > 10000 && x > 1000) && (y > 10000 && x < 1500)`

13. Consider the following code segment

```
int newNum = 0, temp;
int num = k;
while (num > 10)
{
    temp = num % 10;
    num /= 10;
    newNum = newNum * 10 + temp;
}
System.out.print(newNum);
```

num temp newNum

512 2 0

51 1 2

5 20 + 1 = 21

512 → 21

6352 → 253

1000 → 0

Reverses number, but drops last digit

Which is a true statement about the segment?

- ☒ I If $100 \leq \text{num} \leq 1000$ initially, the final value of newNum must be in the range $10 \leq \text{newNum} \leq 100$
- ☒ II There is no initial value of num that will cause an infinite while loop.
- ☒ III If $\text{num} \leq 10$ initially, newNum will have a final value of 0.

- a) I only
- b) II only
- c) III only
- ☒ d) II and III only
- e) I, II, and III

14. Consider the following loop, where n is some positive integer.

```
for (int i = 0; i < n; i += 2)
{
    if (/* test */)
        /* perform some action */
}
```

In terms of n, which Java expression represents the maximum number of times that /* perform some action */ could be executed?

- a) $n / 2$
- ☒ b) $(n + 1) / 2$
- c) n
- d) $n - 1$
- e) $(n - 1) / 2$

Pick some n's

7

8

What i's go into the loop

0, 2, 4, 6 4 times

0, 2, 4, 6 4 times

15. If x is an int where $x = 0$, what will x be after the following loop terminates?

```
while (x < 100)
    x *= 2;
```

- a) 2
- b) 64
- c) 100
- d) 128
- ☒ e) None of the above, this is an infinite loop

16. How many times will the following loop iterate?

```
int x = 10;
while (x > 0) {
    System.out.println(x);
    x--;
}
```

- a) 0 times
- b) 1 time
- c) 9 times
- ☒ d) 10 times
- e) 11 times

10, 9, 8, 7, ..., 1

Big Test Question

What would x be after this loop finishes?

0

x is 0
 $x \neq 100$ changes x to 0

extra question

The next two questions refer to the following code segment:

```
int n = Keyboard.readInt();
int x = 1;
int y = 1;

// Point A

while (n > 2)
{
    x = x + y;

    // Point B

    y = x - y;
    n--;
}

// Point C

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

17. What is printed if the user types in a 6 for n?

- a) 1
- b) 5
- c) 6
- ☒ d) 8
- e) 13

18. What is true about the code?

- ☒ a) x will sometimes be 1 at // Point B
- ☒ b) x will never be 1 at // Point C
- ☒ c) n will never be greater than 2 at // Point A
- ☒ d) n will sometimes be greater than 2 at // Point C
- ☒ e) n will always be greater than 2 at // Point B

19. Given two String variables, s1 and s2, to determine if they are the same length, which of the following conditions would you use?

- a) (s1.equals(s2))
- b) (s1.length() .equals(s2))
- c) (s1.length() .equals(s2.length())
- ☒ d) (s1.length() == s2.length())
- e) length(s1) == length(s2)

Handwritten notes for question 17:

Initial values: x=1, y=1, n=6

Iteration 1 (n=6 > 2): x = 1+1 = 2, y = 2-1 = 1, n = 5

Iteration 2 (n=5 > 2): x = 2+1 = 3, y = 3-1 = 2, n = 4

Iteration 3 (n=4 > 2): x = 3+2 = 5, y = 5-2 = 3, n = 3

Iteration 4 (n=3 > 2): x = 5+3 = 8, y = 8-3 = 5, n = 2

Iteration 5 (n=2 is not > 2): Loop ends. Print x = 8.

20. Given a String, `s`, which is assumed to have at least one character in it, which of the following conditions would determine if the first character of the String is the same as the last character?

- a) `(s.substring(0,1).equals(s.substring(s.length(),s.length()+1)))`
- b) `(s.substring(1,2).equals(s.substring(s.length(), s.length()+1)))`
- c) `(s.substring(0,1).equals(s.substring(s.length()-1, s.length())))`
- d) `(s.substring(0,1).equals(s.substring(s.length()+1, s.length()+2)))`
- e) `(s.substring(0,1).equals(s.substring(last)))`

21. Consider the following code segment

```
int x = 1;
while ( /* missing code */ )
{
    System.out.print(x + " ");
    x = x + 2;
}
```

Consider the following possible replacements for `/* missing code */`

- I. `x < 6`
- II. `x != 6` → infinite loop
- III. `x < 7`

Which of the proposed replacements for `/* missing code */` will cause the code segment to print only the values 1 3 5?

- a) I only
- b) II only
- c) I and II only
- d) I and III only
- e) I, II, and III

Handwritten diagram showing a vertical sequence of numbers: 1, 3, 5, 7, 9. A horizontal line is drawn through the number 5, indicating the sequence of values printed by the code segment.

22. Consider the following code segment

```
int x = 10, y = 0;
while (x > 5)
{
    y = 3;
    while (y < x)
    {
        y *= 2;
        if (y % x == 1)
            y += x;
    }
    x -= 3;
}
System.out.println(x + " " + y);
```

Handwritten notes:

X: 10, 7, 4, 1
Y: 0, 3, 6, 12, 24, 48, 96, 192, 384, 768, 1536, 3072, 6144, 12288, 24576, 49152, 98304, 196608, 393216, 786432, 1572864, 3145728, 6291456, 12582912, 25165824, 50331648, 100663296, 201326592, 402653184, 805306368, 1610612736, 3221225472, 6442450944, 12884901888, 25769803776, 51539607552, 103079215104, 206158430208, 412316860416, 824633720832, 1649267441664, 3298534883328, 6597069766656, 13194139533312, 26388279066624, 52776558133248, 105553116266496, 211106232532992, 422212465065984, 844424930131968, 1688849860263936, 3377699720527872, 6755399441055744, 13510798882111488, 27021597764222976, 54043195528445952, 108086391056891904, 216172782113783808, 432345564227567616, 864691128455135232, 1729382256910270464, 3458764513820540928, 6917529027641081856, 13835058055282163712, 27670116110564327424, 55340232221128654848, 110680464442257309696, 221360928884514619392, 442721857769029238784, 885443715538058477568, 1770887431076116955136, 3541774862152233910272, 7083549724304467820544, 14167099448608935641088, 28334198897217871282176, 56668397794435742564352, 113336795588871485128704, 226673591177742970257408, 453347182355485940514816, 906694364710971881029632, 1813388729421943762059264, 3626777458843887524118528, 7253554917687775048237056, 14507109835375550096474112, 29014219670751100192948224, 58028439341502200385896448, 116056878683004400771792896, 232113757366008801543585792, 464227514732017603087171584, 928455029464035206174343168, 1856910058928070412348686336, 3713820117856140824697372672, 7427640235712281649394745344, 14855280471424563298789490688, 29710560942849126597578981376, 59421121885698253195157962752, 118842243771396506390315925504, 237684487542793012780631851008, 475368975085586025561263702016, 950737950171172051122527404032, 1901475900342344102245054808064, 3802951800684688204490109616128, 7605903601369376408980219232256, 15211807202738752817960438464512, 30423614405477505635920876929024, 60847228810955011271841753858048, 121694457621910022543683507716096, 243388915243820045087367015432192, 486777830487640090174734030864384, 973555660975280180349468061728768, 1947111321950560360698936123457536, 3894222643901120721397872246915072, 7788445287802241442795744493830144, 15576890575604482885591488987660288, 31153781151208965771182977975320576, 62307562302417931542365955950641152, 124615124604835863084731911901282304, 249230249209671726169463823802564608, 498460498419343452338927647605129216, 996920996838686904677855295210258432, 1993841993677373809355710590420516864, 3987683987354747618711421180841033728, 7975367974709495237422842361682067456, 15950735949418990474845684723364134912, 31901471898837980949691369446728269824, 63802943797675961899382738893456539648, 127605887595351923798765477786913079296, 255211775190703847597530955573826158592, 510423550381407695195061911147652317184, 1020847100762815390390123822295304634368, 2041694201525630780780247644590609268736, 4083388403051261561560495289181218537472, 8166776806102523123120990578362437074944, 16333553612205046246241981156724874149888, 32667107224410092492483962313449748299776, 65334214448820184984967924626899496599552, 130668428897640369969935849253798993199104, 261336857795280739939871698507597986398208, 522673715590561479879743397015195972796416, 1045347431181122959759486794030391945592832, 2090694862362245919518973588060783891185664, 4181389724724491839037947176121567782371328, 8362779449448983678075894352243135564742656, 16725558898897967356151788704486271129485312, 33451117797795934712303577408972542258970624, 66902235595591869424607154817945084517941248, 133804471191183738849214309635890169035882496, 267608942382367477698428619271780338071764992, 535217884764734955396857238543560676143529984, 1070435769529469910793714477087121352287059968, 2140871539058939821587428954174242704574119936, 4281743078117879643174857908348485409148239872, 8563486156235759286349715816696970818296479744, 17126972312471518572699431633393941636592959488, 34253944624943037145398863266787883273185918976, 68507889249886074290797726533575766546371837952, 137015778499772148581595453067151533092743675904, 274031556999544297163190906134303066185487351808, 548063113999088594326381812268606132370974703616, 1096126227998177188652763624537212264741949407232, 2192252455996354377305527249074424529483898814464, 4384504911992708754611054498148849058967797628928, 8769009823985417509222108996297698117935595257856, 17538019647970835018444217992595396235871190515712, 35076039295941670036888435985190792471742381031424, 70152078591883340073776871970381584943484762062848, 140304157183766680147553743940763169886969524125696, 280608314367533360295107487881526339773939048251392, 561216628735066720590214975763052679547878096502784, 1122433257470133441180429951526105359095756193005568, 2244866514940266882360859903052210718191512386011136, 4489733029880533764721719806104421436383024772022272, 8979466059761067529443439612208842872766049544044544, 17958932119522135058886879224417685745532099088089088, 35917864239044270117773758448835371491064198176178176, 71835728478088540235547516897670742982128396352356352, 143671456956177080471095033795341485964256792704712704, 287342913912354160942190067590682971928513585409425408, 574685827824708321884380135181365943857027170818850816, 1149371655649416643768760270362731887714054341637701632, 2298743311298833287537520540725463775428108683275403264, 4597486622597666575075041081450927550856217366550806528, 9194973245195333150150082162901855101712434733101613056, 18389946490390666300300164325803710203424869466203226112, 36779892980781332600600328651607420406849738932406452224, 73559785961562665201200657303214840813699477864812904448, 147119571923125330402401314606429681627398955729625808896, 294239143846250660804802629212859363254797911459251617792, 588478287692501321609605258425718726509595822918503235584, 1176956575385002643219210516851437453019191645837006471168, 2353913150770005286438421033702874906038383291674012942336, 4707826301540010572876842067405749812076766583348025884672, 9415652603080021145753684134811499624153533166696051769344, 18831305206160042291507368269622999248307066333392103538688, 37662610412320084583014736539245998496614132666784207077376, 75325220824640169166029473078491996993228265333568414154752, 150650441649280338332058946156983993986456530667136828309504, 301300883298560676664117892313967987972913061334273656619008, 602601766597121353328235784627935975945826122668547313238016, 1205203533194242706656471569255871951891652245337094626476032, 2410407066388485413312943138511743903783304490674189252952064, 4820814132776970826625886277023487807566608981348378505904128, 9641628265553941653251772554046975615133217962696757011808256, 19283256531107883306503545108093951230266435925393514023616512, 38566513062215766613007090216187902460532871850787028047233024, 77133026124431533226014180432375804921065743701574056094466048, 154266052248863066452028360864751609842131487403148112188932096, 308532104497726132904056721729503219684262974806296224377864192, 617064208995452265808113443459006439368525949612592448755728384, 1234128417990904531616226886918012878737051899225184897511456768, 2468256835981809063232453773836025757474103798450369795022913536, 4936513671963618126464907547672051514948207596900739590045827072, 9873027343927236252929815095344103029896415193801479180091654144, 19746054687854472505859630190688206059792830387602958360183308288, 39492109375708945011719260381376412119585660775205916720366616576, 78984218751417890023438520762752824239171321550411833440733233152, 157968437502835780046877041525505648478342643100823666881466466304, 315936875005671560093754083051011296956685286201647333762932932608, 631873750011343120187508166102022593913370572403294667525865865216, 1263747500022686240375016332204045187826741144806589335051731730432, 2527495000045372480750032664408090375653482289613178670103463460864, 5054990000090744961500065328816180751306964579226357340206926921728, 10109980000181489923000130657632361502613929158452714680413853843456, 20219960000362979846000261315264723005227858316905429360827707686912, 40439920000725959692000522630529446010455716633810858721655415373824, 80879840001451919384001045261058892020911433267621717443310830747648, 161759680002903838768002090522117784041822866535243434886621661495296, 323519360005807677536004181044235568083645733070486869773243322990592, 647038720011615355072008362088471136167291466140973739546486645981184, 1294077440023230710144016724176942272334582932281947479092973291962368, 2588154880046461420288033448353884544669165864563894958185946583924736, 5176309760092922840576066896707769089338331729127789916371893167849472, 10352619520185845681152133793415538178676663458255579832743786335698944, 20705239040371691362304267586831076357353326916511159665487572671397888, 41410478080743382724608535173662152714706653833022319330975145342795776, 82820956161486765449217070347324305429413307666044638661950290685591552, 165641912322973530898434140694648610858826615332089277323900581371183104, 331283824645947061796868281389297221717653230664178554647801162742366208, 662567649291894123593736562778594443435306461328357109295602325484732416, 1325135298583788247187473125557188886870612922656714218591204650969464832, 2650270597167576494374946251114377773741225845313428437182409301938929664, 5300541194335152988749892502228755547482451690626856874364818603877859328, 10601082388670305977499785004457511094964903381253713748729637207755718656, 21202164777340611954999570008915022189929806762507427497459274415511437312, 42404329554681223909999140017830044379859613525014854994918548831022874624, 84808659109362447819998280035660088759719227050029709989837097662045749248, 169617318218724895639996560071320177519438454100059419979674195324091498496, 339234636437449791279993120142640355038876908200118839959348390648182996992, 678469272874899582559986240285280710077753816400237679918696781296365993984, 1356938545749799165119972480570561420155507632800475359837393562592731987968, 2713877091499598330239944961141122840311015265600950719674787125185463975936, 5427754182999196660479889922282245680622030531201901439349574250370927951872, 10855508365998393320959779844564491361244061062403802878699148500741855903744, 21711016731996786641919559689128982722488122124807605757398297001483711807488, 43422033463993573283839119378257965444976244249615211514796594002967423614976, 86844066927987146567678238756515930889952488499230423029593188005934847229952, 173688133855974293135356477513031861779904976998460846059186376011869694459904, 347376267711948586270712955026063723559809953996921692118372752023739388919808, 694752535423897172541425910052127447119619907993843384236745504047478777839616, 1389505070847794345082851820104254894239239815987686768473491008094957555679232, 2779010141695588690165703640208509788478479631975373536946982016189915111358464, 5558020283391177380331407280417019576956959263950747073893964032379830222716928, 11116040566782354760662814560834039153913918527901494147787928064759660445433856, 22232081133564709521325629121668078307827837055802988295575856129519320890867712, 44464162267129419042651258243336156615655674111605976591151712259038641781735424, 88928324534258838085302516486672313231311348223211953182303424518077283563470848, 177856649068517676170605032973344626462622696446423906364606849036154567126941696, 355713298137035352341210065946689252925245392892847812729213698072309134253883392, 711426596274070704682420131893378505850490785785695625458427396144618268507766784, 1422853192548141409364840263786757011700981571571391250916854792289236537015533568, 2845706385096282818729680527573514023401963143142782501833709584578473074031067136, 5691412770192565637459361055147028046803926286285565003667419169156946148062134272, 11382825540385131274918722110294056093607852572571130007334838338313892296124268544, 22765651080770262549837444220588112187215705145142260014669676676627784592248537088, 455313021615405250996748884411762243744314102902845200293393533532

23. What will be output by this code segment?

```
for (int i = 5; i > 0; i--)
{
    for (int j = 1; j <= i; j++)
        System.out.print(j*j + " ");
    System.out.println();
}
```

a) 1
1 4
1 4 9
1 4 9 16
1 4 9 16 25

b) 1 4 9 16 25
1 4 9 16
1 4 9
1 4
1

c) 9 7 5 3 1
25 16 9 4
25 16 9
25 16
25

d) 25
25 16
25 16 9
25 16 9 4
25 16 9 4 1

e) 1 4 9 16 25
1 4 9 16 25
1 4 9 16 25
1 4 9 16 25
1 4 9 16 25

Handwritten notes for question 23:

- For the first loop, i starts at 5 and decreases to 1. The number of iterations for each i is written as $\frac{i}{5}, 4, 3, 2, 1$.
- The total number of iterations for the inner loop is calculated as $1+4+9+16+25 = 55$.

24. The following nested loop structure will execute the inner most statement ($x++$) how many times?

```
for(int j = 0; j < 100; j++)
    for(int k = 100; k > 0; k--)
        x++;
```

Handwritten notes for question 24:

- For the first loop, j starts at 0 and increases to 99. The number of iterations for each j is written as $\rightarrow 100 \text{ times}$.
- For the second loop, k starts at 100 and decreases to 1. The number of iterations for each k is written as $\rightarrow 100 \text{ times}$.

a) 100
b) 200
c) 10,000
d) 20,000
e) 1,000,000