TOPIC 6 EXERCISES

Tracing Exercises:

1. Use the following declaration for this problem:

```
     int i,j,k,x;
```

For the three sets of values (i) to (iii), show what is stored in the variable x by each of the following program segments. If x is not assigned a value by the code segment, indicate one by ?.

```
 (i)   i = 1;   j = 2;    k = 3;
 (ii)  i = 7;   j = 10;   k = 4;
 (iii) i = 5;   j = 4;    k = 3;
```

```
(a)  if (j > k)
          x = 2;
     if (i > k)
          x = 3;
```

```
(b)  if (i > j)
          x = 1;
     else
          if (j > k)
               x = 2;
          else
               x = 3;
```

```
(c)  if (i > j)
          x = 5;
     else if (j > k)
          x = 1;
     else if (i > k)
          x = 3;
```

2. For each set of initial values for the variables, evaluate the following series of statements according to Java precedence rules and show what is printed.

```
(i)  a = 6;   b = 7;   c = 3;   d = 2;   e = 16;
(ii) a = 1;   b = 18;  c = 6;   d = 42;  e = 8;
```

```
     int  a,b,c,d,e;
       ...
     if ( ! (a == b && c * c <= 100 || d != e % 8) )
          System.out.println("valid");
     else
          System.out.println("invalid");
     if (1000 > a + b * b * 9 || !(e * d == a / d))
          a += b;
     else
          a -= b;
     System.out.println(a);
```

3. For each set of values for the variables (i) to (iii), show what is printed by code segments (a), (b), and (c). Assume that p, q, and r are all boolean variables whose values are as shown:

```
 (i)    p = true;     q = true;     r = false;
(ii)    p = true;     q = false;    r = true;
(iii)   p = false;    q = false;    r = false;
```

```
(a)  if (p && q || r)                      (b)  if (p || !(q && r))
          System.out.println("true");               System.out.println("true");
     else                                       else
          System.out.println("false");              System.out.println("false");

(c)  if (!p || (q || r) && !q)
          System.out.println("true");
     else
          System.out.println("false");
```

4. For these two programs, substitute each of the following specific conditions one at a time for *condition* in the loop control, then trace the loop. Show what is printed in each case. Will any of these cause an infinite loop or some other error? Here are the five conditions:

```
(i)   (sum <= 12)
(ii)  (num > 4 && sum == num)
(iii) (sum < 12 || num < 16)
(iv)  (num <= 32)
(v)   (sum >= 0)
```

```
(a)public class prob6_4a {
      public static void main(String[] args)
      {
          int num = 2, sum = 0;

          while (condition) {
               num *= 2;
               sum += num;
               System.out.println(num + " " + sum);
          }
          System.out.println(num + " " + sum);
      }
   }
```

```
(b)public class prob6_4b {
      public static void main(String[] args)
      {
          int num = 2, sum = 0;

          do {
               num *= 2;
               sum += num;
               System.out.println(num + " " + sum);
          } while (condition);
          System.out.println(num + " " + sum);
      }
   }
```

5. Show what is printed by each of the following segments. Are (a) and (b) equivalent? What about (c) and (d)?

```
(a)  int a = 1,b = 1,c = 10;       (b) int a = 1,b = 1,c = 10;

     do {                              while (a <= c)  {
          System.out.println(a);            System.out.println(a);
          a += b;                           a += b;
          c++;                              c++;
     } while (a <= c);                 }
     System.out.println(a + " " + b         System.out.println(a + " " + b
          + " " + c);                            + " " + c);

(c)  int a = 1,b = 1,c = 10;       (d) int a = 1,b = 1,c = 10;

     do {                              while (a <= c)  {
          System.out.println(a);            System.out.println(a);
          a += b;                           a += b;
          b++;                              b++;
     } while (a <= c);                 }
     System.out.println(a + " " + b    System.out.println(a + " " + b
          + " " + c);                       + " " + c);
```

6. Show what is printed by the following program. Assume that the set of data consists of these numbers, typed in one at a time in response to the prompt: 12 9 8 6 14 13 10 2.

```
import java.util.Scanner;
public class prob6_6 {
   public static void main(String[] args)
   {
     int i,size;
     Scanner kybd = new Scanner(System.in);

     for (i = 1; i <= 8; i++)  {
          System.out.print("please enter the size: ");
          Size = kybd.nextInt();
          switch(size) {
            case 9:  case 11:  case 13:  case 15:
                 break;  // do nothing for larger odd sizes
            case 2:  case 4:  case 6:  case 8:
                 System.out.println("the dress size is " + size
                      + "--it is small");
                 break;
            case 10:  case 12:
                 System.out.println("the dress size is " + size
                      + "--it is medium");
                 break;
            default:
                 System.out.println("the dress size is " + size
                      + "--it is large");
          }
     }
   kybd.close();
   }
}
```

7. In the following program segment, the method sums() calls the method product(). For each call to a method, identify which is the calling method and which is the called method. Indicate where control returns after the call. Show what is printed, assuming that the data values are 5 and 8.

```
import java.util.Scanner;
public class prob6_7 {
   public static void main(String[] args)
   {
     int x,y,ans;

     Scanner kybd = new Scanner(System.in);

     System.out.println("enter values one at a time for x and y");
     x = kybd,nextInt();
     y = kybd,nextInt();
     ans = sums(x,y);
     System.out.println("x = " + x + " y = " + y + " ans = " + ans);

     kybd.close();
   }

   public static int product(int g, int h)
   {
     if (g * h >= 40)
          return 3;
     else
          return g - h;
   }

   public static int sums(int i, int j)
   {
     int a,b;

     a = product(i,j);
     b = product(a,j);
     return a + b;
   }
}
```

Analysis Exercises

8. Rewrite the following segment using a **switch** statement instead of the nested **if**. Then rewrite it using a cascading nested **if** with full indentation. Finally, rewrite it using a series of **if** statements instead of a nested **if**. Which version is most efficient? Which is easiest to follow? In the nested **if** versions, why can we use simplified **if** conditions? For example, we can use ***if (exam >= 80)*** instead of ***if (exam >= 80 && exam < 90)***.

```
int exam;

if (exam >= 90)
     System.out.println("grade = A");
else if (exam >= 80)
     System.out.println("grade = B");
else if (exam >= 70)
     System.out.println("grade = C");
else if (exam >= 60)
     System.out.println("grade = D");
else
     System.out.println("grade = F");
```

9. Consider the following version of a **switch** statement in which most of the **break** statements have been eliminated.

```
int number;

Scanner kybd = new Scanner(System.in);

System.out.print("type in a number from 2 to 9: ");
number = kybd.nextInt();
switch(number)  {
  case 2:
      System.out.println(number + " is a prime number");
  case 4:
  case 6:
  case 8:
      System.out.println(number + " is an even number");
      break;
  case 3:
  case 5:
  case 7:
      System.out.println(number + " is a prime number");
  case 9:
      System.out.println(number + " is an odd number");
}
```

(a) What is printed if the user types in the following values? 4 7 9 2. Are these correct answers?

(b) If the **break** statements are put back in on each path, what does this version print? Are these correct answers?

10. Are there any differences among the code segments below? Explain. Assume x is an integer variable and that p & q are boolean variables.

```
(a)   (i)   if(!(x == 0))   ...            (ii)   if(x != 0)   ...

(b)   (i)   if(!(p || q))   ...            (ii)   if(!p && !q) ...

(c)   (i)   if(!(p && q))   ...            (ii)   if(!p || !q) ...
```

The equivalences in parts (b) and (c) are called **deMorgan's Laws**.