CS111 Practice Exam Problems for Midterm 1

These are review problems from the book **Introduction to Programming in Java: An Interdisciplinary Approach**, 2nd Edition, Robert Sedgewick and Kevin Wayne, Princeton University.

Review Q&A on chapters:

- 1.1 (p. 9-11)
- 1.2 (p. 37-43)
- 1.3 (p. 78-80)

Problems from section 1.2

1.2.3 – Learning Objectives (3.2f) (3.4a)

a	b	a && b	!(a && b)	(a || b)	!(a || b)
false	**false**	false	true	false	true
false	**true**	false	true	true	false
true	**false**	false	true	true	false
true	**true**	true	false	true	false

(!(a && b) && (a || b))	((a && b) || !(a || b))
false	true
true	false
true	false
false	true

(!(a && b) && (a || b)) || ((a && b) || !(a || b))
true
true
true
true

Since all rows of the whole expressions are true, the whole expression evaluates to true.

1.2.4 – Learning Objectives (3.2f) (3.4a)

Since it's NOT true that a is less than b (!(a<b)) AND it is NOT true that a is greater than b (!(a>b)), the only possible answer is that a is equal to b. Therefore, the simplified expression is a==b.

1.2.5 – Learning Objectives (3.4b)

a	b	a ^ b
false	**false**	false
false	**true**	true
true	**false**	true
true	**true**	false

1.2.7 – Learning Objectives (3.2f) (3.3a)

a) 2bc: when adding an integer to a string, it gives you a string concatenation.

b) 5bc: since the program is evaluating from left to right, it first recognizes 2 and 3 as integers and therefore perform addition of the two values to produce 5. Then, 5 gets concatenated to the string bc and gives you 5bc.

c) 5bc: the parentheses around 2 and 3 gives you the priority to add 2 and 3, but essentially the end result is not different from (b).

d) bc5: it gives you bc5 because the parentheses around 2 and 3 give the program the priority to execute 2+3 first, which give you a 5, and then it will concatenate bc to 5.

e) bc23: it is a little bit surprising because we would expect the output be bc5, but remember that the expression is evaluated from left to right and so "bc"+2 would be evaluated first using string concatenation. Then the next evaluation became "bc2"+ 3, which gives you "bc23" as end result.

1.2.9 – Learning Objectives (3.2f) (3.3a)

a) b: when there are no operators involved, it will simply print out the character b.

b) 197: It evaluates to 197 because when you trying to use + to concatenate character 'b' and 'c', the program would recognize it as an addition operator and it will extract the ASCII value for lowercase b and c (which is 98 and 99, respectively), and print out the sum of the two values.

c) e: It evaluates to e because it first add ‘a’ and 4 together, which produce a sum of 101 (the ASCII for a is 97). Then (char) casts the number to the corresponding character for 101, which is e.

1.2.10 – Learning Objectives (3.2f) (3.3a)

a) 2147483647**:** since a is the maximum integer number you can possibly print, this line of code would simply print out the original value assigned to a.

b) -2147483647**:** since a is already the maximum integer the program can print, when add 1 to a, the end result will be set to Integer.MIN_VALUE by overflow (imagine a wrapped integer set).

c) -2147483647**:** 2-a will give you a normal -2147483645 answer, which is still within the range of the minimum integer.

d) 2147483647**:** The normal answer you will get from -2-a is -2147483649, however, the minimum integer that can be displayed in java is -2147483649, therefore, the end result will be set to Integer.Max Value by underflow.

e) -2: integer overflow. Under the signed binary system, the binary representation for the maximum integer is 0111 1111 1111 1111 1111 1111 1111 1111, when two maximum integers get added together, a integer overflow would occur, the binary representation becomes 1111 1111 1111 1111 1111 1111 1111 1110, in which it equals -2 under the signed binary system. (If adding a third maximum integer (Max_Integer time 3), the value becomes 2147483645, because when -2 gets added to the original representation of the maximum integer number, the resulting integer is within the range, and therefore there are no integer overflow. Therefore, when Max_Integer times an positive odd number, no integer overflow would occur, but if it times an positive even number, an integer overflow would occur and the result is usually equal to the negative value of the positive integer).

f) -4: integer overflow. Same argument as above.

1.2.11 – Learning Objectives (3.2f) (3.3a)

a) 3**.**14159**:** the program printed the original value of a since there are no other operators.
b) 4**.**14159**:** for mixed types of values, Java will automatically convert all values to double and then perform the arithmetic operation.

c) 2: first of all, (int) casts 3.14159 to integer 3 and then the program will divide 8 by 3. Since both values are integers, the program will perform integer division and the result is the quotient, which is 2.
d) 2.5464812403910124: since the type of the values are mixed (8 is an integer and b is a double), Java will automatically convert all values to double and then perform the arithmetic operation.
e) 0: the (int) casts the same answer from (d) to an integer, which is 2. Notice that (int) will not round the answer, but instead truncate everything after the decimal, therefore, even if the original answer is 2.9, it will be truncated to 2 as well.

1.2.17 – Learning Objectives (3.2d)

int a = 1;
a = a + a; a = 1+1 =2
a = a + a; a = 2+2 =4
a = a + a; a = 4+4 =8

boolean a = true;
a = !a; a = false
a = !a; a = true
a = !a; a = false

int a = 2;
a = a * a; a = 2*2 = 4
a = a * a; a = 4*4 = 16
a = a * a; a = 16*16 =256

1.2.14 – Learning Objectives (3.2b) (3.2f) (3.3a) (3.3b) (3.3c) (3.7a)

```
public class EvenDivision {
     public static void main(String args[]) {
              int x = Integer.parseInt(args[0]);
              int y = Integer.parseInt(args[1]);
              boolean isEvenlyDivided = ((x%y)==0 || (y%x)==0);
              System.out.println(isEvenlyDivided);
     }
}
```

1.2.15 – Learning Objectives (3.2b) (3.2f) (3.3a) (3.3b) (3.3c) (3.7a)

```
public class IsTriangle {
     public static void main(String args[]) {
             int a = Integer.parseInt(args[0]);
             int b = Integer.parseInt(args[1]);
             int c = Integer.parseInt(args[2]);
             boolean isTriangle;
             isTriangle=(!( a>=b+c || b>=a+c || c>=a+b));
             System.out.println(isTriangle);
     }
}
```

1.2.18 – Learning Objectives (3.2b) (3.2f) (3.3a) (3.3b) (3.3c) (3.7a) (3.5a)

```
public class Euclidean Distance {
     public static void main(String args[]) {
             int a = Integer.parseInt(args[0]);
             int b = Integer.parseInt(args[1]);
             double a2 = Math.pow(a, 2);
             double b2 = Math.pow(b, 2);
             double eDistance = Math.sqrt(a2+b2);
             System.out.println(eDistance);
     }
}
```

1.2.23 – Learning Objectives (3.2b) (3.2f) (3.3a) (3.3b) (3.3c) (3.7a) (3.5a) (3.4a)

```
public class isInBetween {
     public static void main(String args[]) {
               int m = Integer.parseInt(args[0]);
               int d = Integer.parseInt(args[1]);
               boolean isInBetween = false;
               if(m>=3 && m<=6) {
                    if(m==3) {
                         isInBetween = d>=20 && d<=31;
                    }else if(m==4) {
                         isInBetween = d>=1 && d<=30;
                    }else if(m==5) {
                         isInBetween = d>=1 && d<=31;
                    }else {
                         isInBetween = d>=1 && d<=20;
                    }
               }
               System.out.println(isInBetween);
     }
}
```

1.2.25 – Learning Objectives (3.2b) (3.2f) (3.3a) (3.3b) (3.7a) (3.5a) (3.4a)

```
public class WindChill {
     public static void main(String args[]) {
               double t = Double.parseDouble(args[0]);
               double v = Double.parseDouble(args[1]);
               double w = 35.74 + 0.6215*t + (0.4275*t-
35.75)*Math.pow(v, 0.16);
               System.out.println(w);
     }
}
```

1.2.28 – Learning Objectives (3.2b) (3.2f) (3.3a) (3.3b) (3.7a) (3.4a)

```
public class IsOrdered {
     public static void main(String args[]) {
             double x = Double.parseDouble(args[0]);
             double y = Double.parseDouble(args[1]);
             double z = Double.parseDouble(args[2]);
             boolean isOrdered;
             isOrdered = (x<y && y<z) || (x>y && y>z);
             System.out.println(isOrdered);

     }
}
```

1.2.16 – Learning Objectives (3.1b) (3.2d)

Since the program execute operators from left to right, it will divide r and then times r according to the original formula. To fixed problem, simply put a parenthesis around r * r to give the program the priority to execute r*r first and division the next.

1.2.29 in pseudocode – Learning Objectives (3.2b) (3.2f) (3.3a) (3.3b) (3.3c)

READ month
READ day
READ year
SET y0 TO y-(14-m)/12
SET x TO y0 + y0/4 – y0/100 + y0/400
SET m0 TO m + 12 *((14-m)/12)-2
SET d0 TO (d + x + (31 x m0) / 12)%7

IF d0 equals to Sunday THEN
  DISPLAY Sunday
ELSE IF d0 equals to Monday THEN
  DISPLAY Monday
ELSE IF d0 equals to Tuesday THEN
  DISPLAY Tuesday
ELSE IF d0 equals to Wednesday THEN
  DISPLAY Wednesday
ELSE IF d0 equals to Thursday THEN
  DISPLAY Thursday

```
ELSE IF d0 equals to Friday THEN
  DISPLAY Friday
ELSE
  DISPLAY Saturday
ENDIF
```

Minimum number of operations: 9 when Sunday is displayed
Maximum number of operations: 14 when either Friday or Saturday is displayed

1.2.32 in pseudocode – Learning Objectives (3.2b) (3.2f) (3.3a) (3.3b) (3.3c)

```
READ r
READ g
READ b
IF r equals to 0 AND g equals to 0 AND b equals 0 THEN
        DISPLAY 0
ELSE
        SET w TO the maximum value from r/255, g/255, b/255
        SET c TO (w – (r/255)) / w
        SET m TO (w – (g/255)) / w
        SET y TO (w – (b/255)) /2
        SET k TO 1 – w
        DISPLAY c, m, y, k
```

Minimum number of operations: 7 when IF condition is true
Maximum number of operations: 12 when IF condition is false

Problems from section 1.3

1.3.3 – Learning Objectives (4.1e)

a) Omit “then”
b) Should have parenthesis around a>b
c) This is right
d) Missing a semicolon after c=0

1.3.7 – Learning Objectives (4.1g)

a) 45

b) 1024

c) 15

d) 0: because the program is being evaluated in the following steps:

1. for j + = j++, it can be written as j = j + j++, and the second j and third j are being read first before j++ is performed. So the program stores second j and third j as 0.

2. After storing the values for second and third j, j++ is then being performed, making j = 1.

3. However, the value that will be assigned to the first j in j = j + j ++ has not been calculated yet, and the stored information is j = 0 + 0 (not 1). Therefore, the final j is 0 because the final calculated answer that assigned to j is 0 and this value overrides j = 1 in second step.

1.3.13 – Learning Objectives (4.1d)

m = 987654321
n = 0

1.3.18 – Learning Objectives (4.1d) (4.1g)

d

1.3.44 – Learning Objectives (4.1d)

1st Line: if a is greater than b, swap a and b (after execution, at least a and b are in ascending order (a>b)
2nd Line: if a is greater than c, swap a and c (after execution, at least a and c are in ascending order) (a>c)
3rd Line: if b is greater than c, swap b and c (after execution, b and c are also in ascending order) (b>c)

If (a>b) && (a>c) && (b>c), then it must follow that a>b>c.

1.3.9 – Learning Objectives (4.1a) (4.1f)

```
public class PrintIntegers{
     public static void main(String args[]) {
          for(int i = 1001; i<2000; i+=5) {
               if (i%5==1) {
                    System.out.print(i+" "+(i+1)+" "+(i+2)+"
"+(i+3)+" "+(i+4));
                    System.out.println();
               }
          }
     }
}
```

1.3.16 – Learning Objectives (4.1a) (4.1f)

```
public class PositivePowersof2 {
     public static void main(String args[]) {
          int n = Integer.parseInt(args[0]);
          double i = 0;
          double j = 1;
          if(n>=2) {
               while(Math.pow(2, j)<=n) {
                    i = Math.pow(2, j);
                    System.out.println(i);
                    j++;
               }
          }
     }
}
```

1.3.36 – Learning Objectives (4.1g) (4.1l)

```
public class CountingPrimes {
     public static boolean isPrime(int n) {
          if(n<=1) {
               return false;
          }
          for (int i = 2; i<n; i++) {
               if( n%i == 0) {
                    return false;
               }
          }
          return true;
     }
     public static void main(String args[]) {
          int n = Integer.parseInt(args[0]);
          int count = 0;
          for (int i = 2; i<=n; i++) {
               if(isPrime(i)) {
                    count++;
               }
          }
          System.out.println(count);
     }

}
```

1.3.39 – Learning Objectives (4.1g) (4.1l)

```
public class Sin {

    public static void main(String[] args) {
        double x = Double.parseDouble(args[0]);

        // convert x to an angle between -2 PI and 2 PI
        x = x % (2 * Math.PI);

        // compute the Taylor series approximation
        double term = 1.0;      // ith term = x^i / i!
        double sum  = 0.0;      // sum of first i terms in
taylor series

        for (int i = 1; term != 0.0; i++) {
            term *= (x / i);
            if (i % 4 == 1) sum += term;
            if (i % 4 == 3) sum -= term;
        }
        System.out.println(sum);
    }
}

public class Cos {

    public static void main(String[] args) {
        double x = Double.parseDouble(args[0]);

        // convert x to an angle between -2 PI and 2 PI
        x = x % (2 * Math.PI);

        // compute the Taylor series approximation
        double term = 1.0;      // ith term = x^i / i!
        double sum  = 0.0;      // sum of first i terms in
taylor series

        for (int i = 1; term != 0.0; i++) {
            term *= (x / i);
            if (i % 4 == 0) sum += term;
            if (i % 4 == 2) sum -= term;
        }
        System.out.println(1+sum);
    }
}
```

1.3.43 – Learning Objectives (4.1a) (4.1f) (4.1l)

```
public class MedianOf5 {

    public static void main(String[] args) {
        int a = Integer.parseInt(args[0]);
        int b = Integer.parseInt(args[1]);
        int c = Integer.parseInt(args[2]);
        int d = Integer.parseInt(args[3]);
        int e = Integer.parseInt(args[4]);
        int [] array = {a,b,c,d,e};
        for(int i = 0; i<5 ; i++) {
          for(int j = i+1; j<5; j++) {
               if (array[j]<array[i]) {
                    int temp = array[i];
                    array[i]= array[j];
                    array[j]= temp;
               }
          }
        }
        System.out.println(array[2]);

    }
}
```