SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

01 pizzaParty

```
public boolean pizzaParty(int pizzas, boolean isWeekend) {
  boolean fun = false;

  if (isWeekend) {
    if (40 <= pizzas) {
      fun = true;
    }
  }

  if (!isWeekend) {
    if (40 <= pizzas && pizzas <= 60) {
      fun = true;
    }
  }

  return fun;
}


public boolean pizzaParty(int pizzas, boolean isWeekend) {
  boolean fun = false;

  if ( isWeekend && 40 <= pizzas ) {
    fun = true;
  }

  if ( !isWeekend && (40 <= pizzas && pizzas <= 60) ) {
    fun = true;
  }

  return fun;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

03 squirrelPlay

```
03 public boolean squirrelPlay(int temp, boolean isSummer) {
  boolean play = false;

  if (isSummer) {
    if (60 <= temp && temp <= 100) {
      play = true;
    }
  }

  if (!isSummer) {
    if (60 <= temp && temp <= 90) {
      play = true;
    }
  }

  return play;
}
```

```
03 public boolean squirrelPlay(int temp, boolean isSummer) {
  boolean play = false;

  int upper = 90; // default, NOT isSummer
  if (isSummer) {
    upper = 100;
  }

  if (60 <= temp && temp <= upper) {
    play = true;
  }

  return play;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

04 caughtSpeeding

```
public int caughtSpeeding(int speed, boolean isBirthday) {
  int ticket = 0;

  if (!isBirthday) {
    if (61 <= speed && speed <= 80) {
      ticket = 1;
    }
    if (81 <= speed) {
      ticket = 2;
    }
  }

  if (isBirthday) {
    if (66 <= speed && speed <= 85) {
      ticket = 1;
    }
    if (86 <= speed) {
      ticket = 2;
    }
  }

  return ticket;
}


public int caughtSpeeding(int speed, boolean isBirthday) {
  int ticket = 0;

  int bDay = 0; // a helper variable
  if (isBirthday) {
    bDay = 5;
  }

  if (61 + bDay <= speed && speed <= 80 + bDay) {
    ticket = 1;
  }

  if (81 + bDay <= speed) {
    ticket = 2;
  }

  return ticket;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

05 sortaSum

```
public int sortaSum(int a, int b) {
  int sum = a + b;
  if (10 <= sum && sum <= 19) {
    sum = 20;
  }
  return sum;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

07 love6

```
public boolean love6(int a, int b) {
  boolean answer = false;
  int sum = a + b;
  int diffAB = a - b;
  int diffBA = b - a;
  if (diffBA == 6 || diffAB == 6 || sum == 6 ||
          a == 6 || b == 6) {
    answer = true;
  }
  return answer;
}
```

```
public boolean love6(int a, int b) {
  boolean love = false;
  int sum = a + b;
  int diffAbs = Math.abs(a - b);
  if (a == 6 || b == 6 || sum == 6 || diffAbs == 6) {
    love = true;
  }
  return love;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

08 inOneTo10

```
public boolean in1To10(int n, boolean outsideMode) {
  boolean answer = false;

  if (outsideMode) {
    if (n <= 1 || n >= 10) {
      answer = true;
    }
  }

  if (!outsideMode) {
    if (1 <= n && n <= 10) {
      answer =true;
    }
  }

  return answer;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

09 specialEleven

```
public boolean specialEleven(int n) {
  boolean answer = false;

  if (n % 11 == 0) {
    answer = true;
  }

  if (n % 11 == 1) {
    answer = true;
  }

  return answer;
}

public boolean specialEleven(int n) {
  boolean answer = false;

  if (n % 11 == 0 || n % 11 == 1) {
    answer = true;
  }

  return answer;
}

  // 0 is a multiple.          0 / 11    q 0 r 0
  // 1 is one more than 0.     1 / 11    q 0 r 1
  // 11 is a multiple.         11 / 11   q 1 r 0
  // 12 is one more than 11.   12 / 11   q 1 r 1
  // 22 is a multiple.         22 / 11   q 2 r 0
  // 23 is one more than 22.   12 / 11   q 2 r 1
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

10 more20

```
public boolean more20(int n) {
  boolean answer = false;

  if (n % 20 == 1) {
    answer = true;
  }
  if (n % 20 == 2) {
    answer = true;
  }
  return answer;
}

public boolean more20(int n) {
  boolean answer = false;

  if (n % 20 == 1 || n % 20 == 2) {
    answer = true;
  }

  return answer;
}
```

SOLUTIONS to Selected Problems in
Codingbat Logic-1 Regular Section

13 nearTen

```
public boolean nearTen(int n) {
  boolean within2 = false;
  if (n % 10 == 8) {
    within2 = true;
  }
  if (n % 10 == 9) {
    within2 = true;
  }
  if (n % 10 == 0) {
    within2 = true;
  }
  if (n % 10 == 1) {
    within2 = true;
  }
  if (n % 10 == 2) {
    within2 = true;
  }
  return within2;
}
```

```
public boolean nearTen(int num) {
  int rem = num % 10;
  boolean within2 = false;
  if (rem == 0 || rem == 1 || rem == 2 || rem == 8 || rem == 9) {
    within2 = true;
  }
  return within2;
}
```

```
public boolean nearTen(int n) {
  boolean within2 = false;
  int rem = n % 10;
  if (8 <= rem && rem <= 9) {
    within2 = true;
  }
  if (0 <= rem && rem <= 2) {
    within2 = true;
  }
  return within2;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

14 teenSum

```
public int teenSum(int a, int b) {
  int sum = a + b;
  if (13 <= a && a <= 19) {
    sum = 19;
  }
  if (13 <= b && b <= 19) {
    sum = 19;
  }
  return sum;
}
```

```
public int teenSum(int a, int b) {
  int sum = a + b;
  if  ( (13 <= a && a <=  19) || (13 <= b && b <=  19) ) {
    sum = 19;
  }
  return sum;
}
```

```
// using the HELPER method isTeen (at bottom)
public int teenSum(int a, int b) {
  int sum = a + b;
  if  ( isTeen(a) || isTeen(b) ) {
    sum = 19;
  }
  return sum;
}

public boolean isTeen(int num) {
  boolean teen = false;
  if (13 <= num && num <=  19) {
    teen = true;
  }
  return teen;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

19 twoAsOne

```
public boolean twoAsOne(int a, int b, int c) {
  boolean answer = false;

  if (a == b + c) {
    answer = true;
  }

  if (b == a + c) {
    answer = true;
  }

  if (c == a + b) {
    answer = true;
  }

  return answer;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

20 inOrder

```
public boolean inOrder(int a, int b, int c, boolean bOK) {
  boolean order = false;

  if (!bOK) {
    if (a < b && b < c) {
      order = true;
    }
  }

  if (bOK) {
    if (b < c) {
      order = true;
    }
  }

  return order;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

21 inOrderEqual

```
public boolean inOrderEqual(int a, int b, int c, boolean equalOk) {
  boolean inorder = false;
  if (!equalOk) {
    if (a < b && b < c) {
      inorder = true;
    }
  }

  if (equalOk) {
    if (a <= b && b <= c) {
      inorder = true;
    }
  }
  return inorder;
}
```

SOLUTIONS to Selected Problems in
Codingbat Logic-1 Regular Section

22 lastDigit

```
// SEE Logic-1 BASICS: Using Java's REMAINDER (%) or MOD operation
// # 4: Extracting the rightMost Digit (the digit in the 1s column)

public boolean lastDigit (int a, int b, int c) {
  boolean same = false;

  int lastDigitA = a % 10;
  int lastDigitB = b % 10;
  int lastDigitC = c % 10;

  if (lastDigitA == lastDigitB) {
    same = true;
  }
  if (lastDigitA == lastDigitC) {
    same = true;
  }
  if (lastDigitB == lastDigitC) {
    same = true;
  }

  return same;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

24 withoutDoubles

```
public int withoutDoubles(int die1, int die2, boolean noDoubles) {

  if (noDoubles && die1 == die2) {
    die2 = die2 + 1;
    if (die2 == 7) {
      die2 = 1;
    }
  }

  int sum = die1 + die2;

  return sum;
}
```

SOLUTIONS to Selected Problems in Codingbat Logic-1 Regular Section

29 shareDigit

```
public boolean shareDigit(int a, int b) {

  int digitOnesA = a % 10;
  int digitTensA = (a / 10) % 10;
  int digitOnesB = b % 10;
  int digitTensB = (b / 10) % 10;

  boolean share = false;

  if (digitOnesA == digitOnesB) {
    share = true;
  }
  if (digitOnesA == digitTensB) {
    share = true;
  }
  if (digitTensA == digitOnesB) {
    share = true;
  }
  if (digitTensA == digitTensB) {
    share = true;
  }

  return share;
}
```