Practice Exam #3

SECTION I

Time — 1 hour and 30 minutes
Number of questions — 40
Percent of total grade — 50

1. Assume that the `boolean` variables `x`, `y`, and `z` have been properly declared and initialized. Given that `x` is `true`, `y` is `true`, and `z` is `false`, which of the following expressions will evaluate to `false`?

   (A) `(x && y) || z`
   (B) `(x || y) && z`
   (C) `y || (x && z)`
   (D) `x || (y && z)`
   (E) `x && (y || z)`

2. Given two initialized `int` variables `a` and `b`, which of the following expressions evaluates to the same result as `2*(1 + (3*b)/a)` for all positive values of `a` and `b`?

   (A) `2 + 6*b/a`
   (B) `2*(1 + b/a*3)`
   (C) `2 + 2*(b*3/a)`
   (D) All of the above
   (E) None of the above

3. Consider the following statement.

```
System.out.println("\\\"hello\"".substring(1));
```

   What is printed when the statement is executed?

   (A) `hello`
   (B) `\hello`
   (C) `\\hello`
   (D) `"hello"`
   (E) `\\"hello"`

4. Consider the following code segment.

```
String[] a = {"A", "B"};
String[] b = a;
String[] c = {a[0], a[1]};

String letter = a[0]; a[0] = a[1]; a[1] = letter;

System.out.println(a[0] + a[1] + b[0] + b[1] + c[0] + c[1]);
```

What is printed when the code segment is executed?

(A) BAABAB
(B) BAABBA
(C) BABAAB
(D) BABABA
(E) BBBBBB

5. Given

```
String s = "ALASKA";
int index = s.substring(1, 4).indexOf("A");
```

what is the value of `index`?

(A) 0
(B) 1
(C) 2
(D) 5
(E) -1

6. What is the value of `v[4]` after the following code segment is executed?

```
int d = 1;
int[] v = {1, 1, 1, 1, 1};

for (int i = 0; i < v.length; i++)
{
  d *= 2;
  v[i] += d;
}
```

(A) 16
(B) 32
(C) 33
(D) 64
(E) 65

7. Consider the following method `fun2`.

```
public int fun2(int x, int y)
{
  y -= x;
  return y;
}
```

What are the values of the variables `a` and `b` after the following code segment is executed?

```
int a = 3, b = 7;
b = fun2(a, b);
a = fun2(b, a);
```

(A) `-1` and `4`
(B) `-4` and `7`
(C) `-4` and `4`
(D) `3` and `7`
(E) `3` and `4`

8. Consider the following method.

```
public void splat(String s)
{
  if (s.length() < 8)
    splat(s + s);
  System.out.println(s);
}
```

What is printed when `splat("**")` is called?

(A)
```
**
```

(B)
```
****
```

(C)
```
********
```

(D)
```
********
**
```

(E)
```
********
****
**
```

Questions 9 and 10 refer to the following method.

```
public void printVals(String[] items, int k)
{
  if (k > 1)
  {
    printVals(items, k - 1);
    System.out.print(items[k] + " ");
    printVals(items, k - 2);
  }
}
```

Consider the following code segment.

```
String[] names = {"Pat", "Joe", "Ann", "Cal", "Amy"};
printVals(names, names.length - 1);
```

9. What is printed when the code segment is executed?

(A) `Ann Cal Amy Ann`
(B) `Ann Cal Amy Cal Ann`
(C) `Ann Cal Joe Amy Joe Ann`
(D) `Joe Ann Cal Joe Amy Joe Ann`
(E) `Joe Ann Pat Cal Joe Amy Joe Ann Pat`

10. How many calls to `printVals` are made, including the original call, when the code segment is executed?

(A) 3
(B) 5
(C) 7
(D) 8
(E) 9

11. What is the result when the following code segment is compiled and executed?

```
int m = 4, n = 5;
double d = Math.sqrt((m + n)/2);
System.out.println(d);
```

(A) Syntax error "sqrt(double) in java.lang.Math cannot be applied to int"
(B) `1.5` is printed
(C) `2.0` is printed
(D) `2.1213203435596424` is printed
(E) `ArithmeticException`

12. Suppose a class `MyTest` has an instance variable `score`:

```
private int score = 0;
```

Which of the following could serve as a constructor in this class?

(A) `public MyTest() { }`
(B) `public int MyTest() { return score; }`
(C) `public MyTest(int s) { this = s; }`
(D) `public void MyTest(int s) { score = s; }`
(E) `public MyTest MyTest(int s) { score = s; return this; }`

13. Consider the following class `Athlete`.

```
public class Athlete
{
  private int numMedals;

  public int getRank() { return numMedals; }

  public int compareTo(Athlete other)
  {
    // return numMedals - other.numMedals;
    return getRank() - other.getRank();
  }

  < constructors and other methods not shown >
}
```

As you can see, the programmer has commented out direct references to `Athlete`'s instance variable `numMedals` in the `compareTo` method and replaced them with calls to the `getRank` method. What is the most compelling reason for doing this?

(A) To correct a syntax error: being private, neither `numMedals` nor `other.numMedals` is directly accessible in the method's code
(B) To correct a syntax error: being private, `other.numMedals` is not directly accessible in the method's code (`numMedals` is replaced with `getRank()` for consistency)
(C) To avoid possible problems later: if `other` happens to be an object of a subclass of `Athlete` in which `numMedals` is not used in calculating the rank, the original code would fail
(D) To improve run-time efficiency
(E) To achieve better encapsulation of the `Athlete` class

Questions 14 and 15 refer to the following classes.

```
public class Point
{
  private int x, y;

  public Point(int _x, int _y) { x = _x; y = _y; }

  public int getX() { return x; }
  public int getY() { return y; }

  public void move(int dx, int dy) { x += dx; y += dy; }
}

public class Polygon
{
  private ArrayList<Point> vertices;

  public Polygon()
  {
    vertices = new ArrayList<Point>();
  }

  public void add(Point p) { vertices.add(p); }

  /** Returns the x-coordinate of the k-th vertex
   *  (counting from 0)
   */
  public int getX(int k)
  { return < missing expression >; }

  /** Moves every vertex of the polygon horizontally by dx
   *  and vertically by dy
   */
  public void move(int dx, int dy)
  { < missing code > }
}
```

14. Which of the following could replace < *missing expression* > in the `getX` method of the `Polygon` class?

(A) `vertices[k].x`
(B) `vertices.getX(k)`
(C) `vertices.get(k).x`
(D) `vertices.get(k).getX()`
(E) `vertices.getX().get(k)`

15. Which of the following could replace < *missing code* > in `Polygon`'s `move` method for it to work as described in the comment for this method?

I.
```
for (int k = 0; k < vertices.size(); k++)
{
  Point p = vertices.get(k);
  p.x += dx;
  p.y += dy;
}
```

II.
```
for (int k = 0; k < vertices.size(); k++)
  vertices.get(k).move(dx, dy);
```

III.
```
for (Point p : vertices)
  p.move(dx, dy);
```

(A) I only
(B) II only
(C) I and II only
(D) II and III only
(E) I, II, and III

16. Consider the following method from `ClassX`.

```
private int modXY(int x, int y)
{
  r = x / y;
  return x % y;
}
```

If `ClassX` compiles with no errors, which of the following must be true?

I. `r` must have the type `double`.
II. `r` is not a local variable in the `modXY` method.
III. `r` must be a static variable in `ClassX`.

(A) I only
(B) II only
(C) I and II only
(D) II and III only
(E) I, II, and III

17. Consider the following code segment.

```
int[] factors = {2, 3, 4, 7, 2, 5};
int product = 1;

for (int i = 1; i < factors.length; i += 2)
{
  product *= (factors[i] % factors[i - 1]);
}
```

What is the value of `product` after the code segment is executed?

(A) 0
(B) 1
(C) 2
(D) 3
(E) 5

18. Consider the following class.

```
public class BuddyList
{
  /** Contains the names of buddies. */
  private ArrayList<String> buddies;

  public ArrayList<String> getBuddies()
  {
    return buddies;
  }

  < constructors, other fields and methods not shown >
}
```

If `BuddyList myFriends` is declared and initialized in some other class, a client of `BuddyList`, which of the following correctly assigns to `name` the name of the first buddy in the `myFriends` list?

I. `String name = myFriends.buddies[0];`
II. `String name = myFriends.buddies.get(0);`
III. `String name = myFriends.getBuddies().get(0);`

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) II and III only

19. Consider the following method.

```
/** Returns the location of the target value
 *  in the array a, or -1 if not found.
 *  Precondition: a[0] ... a[a.length - 1] are
 *                sorted in ascending order.
 */
public static int search(int[] a, int target)
{
  int first = 0;
  int last = a.length - 1;

  while (first <= last)
  {
    int middle = (first + last) / 2;
    if (target == a[middle])
      return middle;
    else if (target < a[middle])
      last = middle;
    else
      first = middle;
  }
  return -1;
}
```

This method fails to work as intended under certain conditions. If the array has five elements with the values 3, 4, 35, 42, 51, which of the following values of `target` would make this method fail?

(A) 3
(B) 4
(C) 35
(D) 42
(E) 51

20. The method

```
private void transpose(int[][] m)
{
  < implementation not shown >
}
```

flips the elements of `m` symmetrically over the diagonal. For example:

```
1 2 3          transpose          1 4 7
4 5 6    ------------------>       2 5 8
7 8 9                               3 6 9
```

Which of the following implementations of `transpose` will work as intended?

I.
```
for (int r = 0; r < m.length; r++)
{
  for (int c = 0; c < m[0].length; c++)
  {
    int temp = m[r][c];
    m[r][c] = m[c][r];
    m[c][r] = temp;
  }
}
```

II.
```
for (int c = m[0].length - 1; c > 0; c--)
{
  for (int r = c-1; r >= 0; r--)
  {
    int temp = m[r][c];
    m[r][c] = m[c][r];
    m[c][r] = temp;
  }
}
```

III.
```
for (int c = 0; c < m[0].length - 1; c++)
{
  for (int r = c + 1; r < m.length; r++)
  {
    int temp = m[r][c];
    m[r][c] = m[c][r];
    m[c][r] = temp;
  }
}
```

(A) I only
(B) II only
(C) I and II only
(D) II and III only
(E) I, II, and III

21. Consider the following code segment.

```
String s = "ban";

ArrayList<String> words = new ArrayList<String>();
words.add(s);
words.add(s.substring(1));
words.add(s.substring(1, 2));

String total = "";

for (String w : words)
{
  total += w;
}
System.out.print(total.indexOf("an"));
```

What is printed when the code segment is executed?

(A) 1
(B) 2
(C) 3
(D) ana
(E) banana

22. Consider the following method.

```
public double goFigure(int n)
{
  n %= 7;
  return (double)(12 / n);
}
```

What is printed when the following code segment is executed?

```
int n = 12;
System.out.print(goFigure(n));
System.out.println(" " + n);
```

(A) 2.0 12
(B) 2.4 12
(C) 2.0 5
(D) 2.4 5
(E) 2.4 6

23. Consider the following code segment.

```
int n = 1, d = 1, s = 1;

while (n <= 5)
{
  System.out.print(s + " ");
  n++;
  d += 2;
  s += d;
}
System.out.println();
```

What is printed when the code segment is executed?

(A) 1 3 6 10 15
(B) 1 4 9 16 25
(C) 1 4 7 10 13
(D) 1 4 9 16
(E) 1 3 5 7

24. Consider the following classes.

```
public class A
{
  public A() { methodOne(); }

  public void methodOne() { System.out.print("A"); }
}

public class B extends A
{
  public B() { System.out.print("*"); }

  public void methodOne() { System.out.print("B"); }
}
```

What is printed when the following statement is executed?

```
A obj = new B();
```

(A) *
(B) *A
(C) *B
(D) A*
(E) B*

25. Consider the following class.

```
public class Rectangle
{
  private int width, height;

  public Rectangle(int w, int h)
  {
    width = w;
    height = h;
  }

  public int getArea() { return width * height; }

  < other methods not shown >
}
```

Suppose this class also overrides `Object`'s `equals` method in such a way that the method returns `true` for `Rectangle` objects with the same area. Which of the following `equals` methods will accomplish this?

(A)
```
public int equals(int area)
{
  return getArea() - area;
}
```

(B)
```
public boolean equals(int area)
{
  return getArea() == area;
}
```

(C)
```
public boolean equals(Rectangle other)
{
  return getArea() == other.getArea();
}
```

(D)
```
public boolean equals(Object other)
{
  return this.getArea() == other.getArea();
}
```

(E)
```
public boolean equals(Object other)
{
  return getArea() == ((Rectangle)other).getArea();
}
```

26. Consider the following incomplete method `cutToAverage`.

```
public double cutToAverage(double[] amps)
{
  double avg = 0.0;

  < missing code >

  return avg;
}
```

It first finds the average `avg` of the values in an array, then replaces every element that exceeds `avg` with `avg`. `cutToAverage` returns `avg`. Which of the following could replace < *missing code* > so that `cutToAverage` works as intended?

I.
```
for (double x : amps)
  avg += x;
avg /= amps.length;
for (double x : amps)
  if (x > avg)
    x = avg;
```

II.
```
for (int k = 0; k < amps.length; k++)
  avg += amps[k];
avg /= amps.length;
for (int k = 0; k < amps.length; k++)
  if (amps[k] > avg)
    amps[k] = avg;
```

III.
```
for (double x : amps)
  avg += x;
avg /= amps.length;
for (int k = amps.length - 1; k >= 0; k--)
  if (amps[k] > avg)
    amps[k] = avg;
```

(A) I only
(B) II only
(C) I and II only
(D) II and III only
(E) I, II, and III

27. Consider the following classes.

```
public class Lunch extends Meal
{
  public Lunch() { }

  <other constructors, methods, and instance variables not shown>
}

public class Pizza extends Lunch
{
  public Pizza() { }

  <other constructors, methods, and instance variables not shown>
}
```

Which of the following statements will cause a compile-time error?

(A) `Lunch pizza = new Lunch();`
(C) `Lunch pizza = new Pizza();`
(B) `Meal pizza = new Lunch();`
(D) `Meal pizza = new Pizza();`
(E) None of the above

28. Consider the following method.

```
public int locate(String str, String oneLetter)
{
  int j = 0;
  while (j < str.length() &&
         str.substring(j, j+1).compareTo(oneLetter) < 0)
  {
    j++;
  }
  return j;
}
```

Which of the following must be true when the `while` loop terminates?

(A) `j == str.length()`
(B) `str.substring(j, j+1) >= 0`
(C)
```
j <= str.length() ||
     str.substring(j, j+1).compareTo(oneLetter) > 0
```
(D)
```
j >= str.length() ||
     str.substring(j, j+1).compareTo(oneLetter) >= 0
```
(E)
```
j >= str.length() &&
     str.substring(j, j+1).compareTo(oneLetter) >= 0
```

Questions 29 and 30 refer to the following implementation of Mergesort.

```
public class Mergesort
{
  /** Returns a new array that holds the values
   *  arr[m], arr[m+1], ... arr[n] arranged in ascending order
   *  Precondition: 0 <= m <= n < arr.length
   */
  public static int[] sort(int[] arr, int m, int n)
  {
    int[] result = new int[n - m + 1];

    if (m == n)
    {
      result[0] = arr[m];
    }
    else
    {
      int mid = (n + m) / 2;
      int[] result1 = sort(arr, m, mid);
      int[] result2 = sort(arr, mid + 1, n);
      result = merge(result1, result2);
    }

    return result;
  }

  /** Merges arr1 and arr2 in ascending order and returns the
   *  resulting array.
   *  Precondition: arr1 and arr2 are sorted in ascending order
   */
  private static int[] merge(int[] arr1, int[] arr2)
  { < implementation not shown > }
}
```

29. If `int[] arr` holds eight values and `Mergesort(arr, 0, 7)` is called, how many times in total will `Mergesort`'s `merge` method be called?

(A) 1
(B) 3
(C) 7
(D) 8
(E) 15

30. If `Mergesort.sort(arr, 0, 999)` takes on average 40 ms and `Mergesort.merge(arr1, arr2)` takes on average `0.01*(arr1.length + arr2.length)`, what is the average run time for `Mergesort.sort(arr, 0, 1999)`?

(A) 50 ms
(B) 100 ms
(C) 160 ms
(D) 170 ms
(E) 180 ms

31. Consider the following code segment.

```
ArrayList<String> xyz = new ArrayList<String>();
xyz.add("X");
xyz.add("Y");
xyz.add("Z");

int count = 0;
for (String s1 : xyz)
{
  for (String s2 : xyz)
  {
    if (s1.equals(s2))
    {
      count++;
    }
  }
}

System.out.print(count);
```

What is the result when the code segment is compiled/executed?

(A) 0 is printed
(B) 1 is printed
(C) 3 is printed
(D) `NullPointerException`
(E) Syntax error

32. A two-dimensional array `image` holds brightness values for pixels (picture elements) in an image. The brightness values range from 0 to 255. Consider the following method.

```
public int findMax(int[][] image)
{
  int[] count = new int[256];
  int i, iMax = 0;

  for (int r = 0; r < image.length; r++)
  {
    for (int c = 0; c < image[0].length; c++)
    {
      i = image[r][c];
      count[i]++;
    }
  }

  for (i = 1; i < 256; i++)
  {
    if (count[i] > count[iMax])
      iMax = i;
  }

  return iMax;
}
```

What does this method compute?

(A) The column with the largest sum of brightness values in `image`
(B) The maximum brightness value for all pixels in `image`
(C) The most frequent brightness value in `image`
(D) The maximum sum of brightness values in any 256-by-256 square in `image`
(E) The maximum sum of brightness values in any 256 consecutive rows in `image`

33. What is the result when the following code segment is compiled/executed?

```
String a = "A";
String b = a;          // Line 1
String c = a + 1;      // Line 2
String d = b + "1";

System.out.println((c == d) + " " + c.equals(d));
```

(A) `false false` is printed
(B) `false true` is printed
(C) `true true` is printed
(D) Syntax error on Line 1
(E) Syntax error on Line 2

34. Consider the following class.

```
public class BankAccount
{
  private int balance;

  public BankAccount() { balance = 0; }
  public BankAccount(int amt) { balance = amt; }

  public int getBalance() { return balance; }
  public void makeDeposit(int amt) { balance += amt; }
}
```

What is printed when the following code segment in a client class is executed?

```
ArrayList<BankAccount> bank = new ArrayList<BankAccount>();
bank.add(new BankAccount());
bank.add(new BankAccount(5));
bank.add(new BankAccount(10));
bank.add(new BankAccount(15));

for (BankAccount customer : bank)
  customer.makeDeposit(10);
int total = 0;
for (BankAccount customer : bank)
  total += customer.getBalance();
System.out.println(total);
```

(A) `0`
(B) `30`
(C) `40`
(D) `60`
(E) `70`

35. The statement

```
System.out.println(Integer.MAX_VALUE);
```

prints `2147483647`, which is $2^{31}-1$. What is the result when

```
System.out.println(2*Integer.MAX_VALUE + 2);
```

is compiled/executed?

(A) `0` is printed
(B) `4294967296`, which is 2^{32}, is printed
(C) `ArithmeticException`
(D) Syntax error: arithmetic overflow
(E) Syntax error: cannot apply the `+` operator to an `Integer` and an `int`

Questions 36-38 refer to the following classes.

```
public class Party
{
  private ArrayList<String> theGuests;

  public Party() { theGuests = null; }

  public Party(ArrayList<String> guests) { theGuests = guests; }

  public void setGuests(ArrayList<String> guests)
  { theGuests = guests; }

  public String toString()
  { /* implementation not shown */ }
}

public class BDayParty extends Party
{
  private String theName;

  public BDayParty(String name, ArrayList<String> guests)
  { /* implementation not shown */ }

  public String getName() { return theName; }

  < other methods not shown >
}
```

36. Given

```
ArrayList<String> guests = new ArrayList<String>();
guests.add("Amal");
guests.add("Ben");
guests.add("Candy");
```

which of the following declarations is NOT valid?

(A) `Party[] celebrations = new Party[2];`

(B)
```
Party[] celebrations =
  {new Party(guests), new Party()};
```

(C)
```
BDayParty[] celebrations =
  {new BDayParty("Malika", guests), new Party(guests)};
```

(D)
```
BDayParty[] celebrations =
  {new BDayParty("Ethan", guests),
   new BDayParty("Henry", guests)};
```

(E) All of the above are valid.

37. Consider the following constructor in the `BDayParty` class.

```
public BDayParty(String name, ArrayList guests)
{
  < missing statement >
  theName = name;
}
```

Which of the following statements can replace < *missing statement* > in this constructor?

I. `theGuests = guests;`

II. `super(guests);`

III. `setGuests(guests);`

(A) I only
(B) II only
(C) I and II only
(D) II and III only
(E) I, II, and III

38. Suppose we have added the following methods to the `Party` class:

```
public String getOccasion() { return null; }
public String getMessage() { return "Happy"; }
public String greetings() { return getMessage() + " "
                                  + getOccasion(); }
```

and

```
BDayParty birthday = new BDayParty("Aaron", guests);
System.out.println(birthday.greetings());
```

prints

```
Happy Birthday Aaron
```

Which of the following is the smallest set of `Party` methods that would have to be overridden in the `BDayParty` class to make this happen?

(A) None
(B) `getOccasion`
(C) `getMessage`
(D) `getOccasion` and `getMessage`
(E) `getOccasion`, `getMessage`, and `greetings`

39. Consider the following code segment.

```
ArrayList<Integer> numbers = new ArrayList<Integer>();
numbers.add(0);
numbers.add(1);

for (int k = 1; k <= 3; k++)
{
  int n = numbers.size();
  for (int i = 0; i < n; i++)
    numbers.add(numbers.get(i) + 1);
}

int n = numbers.size();
System.out.println(n + " " + numbers.get(n - 1));
```

What is printed when the code segment is executed?

(A) 6 3
(B) 8 3
(C) 8 4
(D) 16 3
(E) 16 4

40. Consider the following method.

```
public int randomPoints(int n)
{
  return (int)(n * Math.random()) + 1;
}
```

Which of the following outputs is NOT possible when the statement

```
System.out.println(randomPoints(3) + randomPoints(3));
```

is executed?

(A) 2
(B) 3
(C) 4
(D) 6
(E) All of the above are possible

Practice Exam #3

SECTION II

Time — 1 hour and 30 minutes
Number of questions — 4
Percent of total grade — 50

1. The class `Bus` is a simplified model of a bus schedule at a particular bus stop. You will write a constructor and a method of this class.

```
public class Bus
{
  /** Starting and ending times of the bus schedule
      (in minutes since midnight) at a particular bus stop */
  private int startTime, endTime;

  /** The interval between two consecutive buses (in minutes) */
  private int runInterval;

  /** Converts a string that represents time in the 24-hour
   *  format into minutes since midnight and returns the
   *  result.
   */
  private int toMinutes(String time)
  { /* implementation not shown */ }

  /** Constructs a Bus object with given starting and
   *  ending times and a given interval between two
   *  consecutive buses. The start and end strings represent
   *  time in the 24-hour format.
   */
  public Bus(String start, String end, int mins)
  { /* to be implemented in part (a) */ }

  /** Returns the wait time, in minutes, for the next bus
   *  from the given time, as described in part (b).
   */
  public int waitTime(String time)
  { /* to be implemented in part (b) */ }
}
```

(a) Write the constructor of the `Bus` class. Use the provided helper method `toMinutes` to convert the parameters `String start` and `String end`, which represent time in the 24-hour format, into minutes since midnight. For example, `toMinutes("06:30")` returns 390 (because $6 \cdot 60 + 30 = 390$) and `toMinutes("18:30")` returns 1110, so if a `Bus` object is constructed with parameters `"06:30"`, `"18:30"`, and `20`, its instance variables `startTime`, `endTime`, and `runInterval` get the values 390, 1110, and 20, respectively.

Complete the constructor of the `Bus` class below.

```
/** Constructs a Bus object with given starting and
 *  ending times and a given interval between two
 *  consecutive buses. The start and end strings represent
 *  time in the 24-hour format.
 */
public Bus(String start, String end, int mins)
```

(b) The method `waitTime` of the `Bus` class takes one parameter, a string that represents the current time in the 24-hour format. The method returns the wait time, in minutes, for the next bus. If the current time exceeds the time of that day's final bus, the method returns -1. For example, if a `bus` object is constructed like this —

```
Bus bus = new Bus("6:00", "18:30", 20);
```

— then `bus.waitTime(time)` returns the following values:

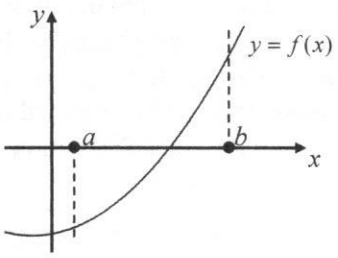
`time`	Time in minutes since midnight	Next bus	`bus.waitTime(time)`
`"5:20"`	320	`"6:00"` => 360	40
`"7:10"`	430	`"7:20"` => 440	10
`"7:20"`	440	`"7:20"` => 440	0
`"18:25"`	1105	Last bus left at 18:20	-1
`"18:40"`	1120	No more buses	-1

Complete the `waitTime` method below. Call the `toMinutes` method to convert the `time` parameter into minutes since midnight.

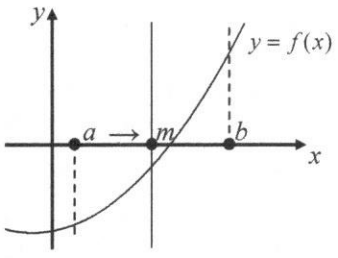
```
/** Returns the wait time for the next bus from a given
 *  time. The parameter time is a string that represents
 *  time in the 24-hour format. If time exceeds the time
 *  of the final bus, the method returns -1.
 */
public int waitTime(String time)
```

For additional practice, implement the `toMinutes` method and provide a `main` method to test the code you wrote in Parts (a) and (b).

2. The class `EquationSolver` helps find an approximate solution for an equation $f(x) = 0$. (The function $f(x)$ is defined in a separate class `Fun`.) `EquationSolver`'s method `solve` looks for a solution on the specified interval $[a, b]$. It is known that the function f is continuous on $[a, b]$, $f(a) < 0$, and $f(b) \geq 0$. For example:



The `solve` method uses a "divide-and-conquer" algorithm to find the approximate solution. It takes the midpoint of the interval $[a, b]$, $m = \frac{a+b}{2}$, and examines $f(m)$. If $f(m) < 0$, a is set to m; otherwise b is set to m.



The process continues until the value of $b - a$ becomes smaller than the specified precision value. Then `solve` returns the midpoint of the last interval $[a, b]$.

`EquationSolver`'s constructor takes one `double` parameter, the desired precision of the solution. The `solve` method takes three parameters: an object of the type `Fun` (or of a subclass of `Fun`) that has a public method `double f(double x)`, and `double` values `a` and `b`. For example, if `EquationSolver` were constructed with the precision constant 0.000001 and $f(x)$ were defined as

```
public double f(double x)
{
  return Math.pow(x, 4) - 0.3;
}
```

(that is, $y = x^4 - 0.3$), then `solve` would return 0.7400828301906586, which is pretty close to the "exact" solution $0.3^{0.25} \approx 0.7400828044922853$.

Assume that the precondition $f(a) < 0$ and $f(b) \geq 0$ in the `solve` method is satisfied — no need to check for it.

Write the complete `EquationSolver` class, including a constructor that initializes the precision constant to a given value and the `solve` method described above. Your implementation must use the "divide-and-conquer" algorithm described above. **Do not use arrays, `ArrayLists`, or other data structures in your solution.**

For additional practice, supply a class `Fun` and define the `f(x)` method in it to return `Math.pow(x, 4) - r`, where `r` is a random number between 0 and 1. Also add a method that returns the "exact" solution, `Math.pow(r, 0.25)`. Test your `EquationSolver` class by calling the `solve` method several times and comparing the solutions produced by `solve` with the "exact" solutions.*

* Ordinarily, `Fun` would be an interface that specifies one method, `double f(double x)`, with different classes implementing that interface, but Java interfaces are not tested on AP CSA exams.

3. Apflix recommends movies to customers based on their previous selections. Clearly some movie matching mechanism is employed. A movie is described by a set of predefined features, such as genre, rating, director, lead actress, lead actor, and so on. A movie is represented by an object of the class `Movie`:

```
public class Movie
{
  /** Returns the list of features for this movie. */
  public ArrayList<String> getFeatures()
  {  /* implementation not shown */ }

  /** Returns the likeness score between this movie and other,
   *  as explained in part (a).
  public double likenessScore(Movie other)
  {  /* to be implemented in part(a) */  }

  /* fields, constructors, and other methods not shown */
}
```

The `getFeatures` method of the `Movie` class returns an `ArrayList<String>` whose elements represent the features of a movie. The sizes of the features lists are the same for all movies.

(a) The likeness score of two movies is defined as the number of matching features divided by the total number of features. For example:

`movie1` features	`["COM", "ROM", "PAL", "AFF", "PG"]`
`movie2` features	`["DRA", "ROM", "PAL", "HOP", "PG"]`
Total number of features	5
Number of matching features	3
`movie1.likenessScore(movie2)`	3/5 = 0.6

Complete the method `likenessScore` below.

```
/** Returns the likeness score between this movie and other.
 *  Precondition: The length of the features list is the
 *                same in this and other, and the features
 *                are listed in the same order in both
 *                movies.
 */
public double likenessScore(Movie other)
```

(b) An Apflix subscriber has a list of movies she has watched recently. Some of the movies on that list may not fit well with the rest (for example, a movie may have been ordered for a friend or a family member). Apflix's marketing department wishes to detect and remove such "outliers" from each subscriber's movie list.

An Apflix subscriber is represented by the class `ApflixSubscriber`:

```
public class ApflixSubscriber
{
  /** The list of movies watched recently by this subscriber */
  private ArrayList<Movie> movies;

  /** Returns an array of fit coefficients that correspond to
   *  movies in the subscriber's list; a coefficient for a
   *  movie indicates how well that movie fits with the rest
   *  of the movies in the list.
   */
  public double[] getFitCoefficients()
  {  /* implementation not shown */ }

  /** Removes outliers from the movies list, as explained
   *  below.
   */
  public void removeOutliers()
  {  /* to be implemented in part (b) */  }

  /* constructors and other fields and methods not shown */
}
```

The method `getFitCoefficients` returns an array of values that measure how well each movie in the list `movies` fits with the rest. The size of the array returned by `getFitCoefficients` is the same as the size of the `movies` list. The *k*-th element in the returned array is equal to the fit coefficient for the *k*-th movie.

Write the method `removeOutliers`. This method obtains an array of the fit coefficients for the `movies` list, calculates their average, and uses half of that value as a threshold. It then removes from the `movies` list all the elements whose fit coefficient is less than the threshold.

Complete the method `removeOutliers` below.

```
/** Obtains the fit coefficients for the movies list,
 *  calculates their average, and removes from the
 *  movies list all the elements whose fit coefficient
 *  is less than half the average.
 *  Precondition: The size of the movies list is >= 2.
 */
public void removeOutliers()
```

For additional practice, complete the `Movie` class, then implement the `getFitCoefficients` method of the `ApflixSubscriber` class. The fit coefficient for a movie *M* is defined as the average of the likeness scores between *M* and all the other movies in the list. Make sure that the likeness score for any pair of movies from the list is computed only once, but do not use any temporary arrays or other data structures.

4. A ski resort architect is using a program to help her lay out trails. The slope is represented as a two-dimensional array of integers, in which each element represents a location on the slope and holds the altitude at that location. The slope is modeled by the class `SkiSlope`.

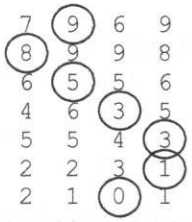
```
public class SkiSlope
{
  /** The altitudes of locations on the slope */
  private int[][] alts;

  /** Returns the index of the element in the top row of alts
   *  that holds the greatest value of all the alts values in
   *  the top row.
   */
  public int summitLocation()
  { /* to be implemented in part (a) */ }

  /** Builds and returns the "steepest trail" on the slope,
   *  as explained in part (b).
   */
  public ArrayList<Integer> findSteepestTrail()
  { /* to be implemented in part (b) */}

  /* constructors and other fields and methods not shown */
}
```

(a) The summit of the slope is the location in the top (index 0) row of the `alts` array that has the greatest value in that row. If several locations have the same greatest altitude, the summit is assumed to be the leftmost of them. Write the method `summitLocation` that finds the summit of the slope and returns the column index of the summit.

Complete the method `summitLocation` below.

```
/** Returns the index of the element in the top row of alts
 *  that holds the greatest value of all the alts values in
 *  the top row.
 */
public int summitLocation()
```

(b) Write a method `findSteepestTrail` that builds and returns the "steepest trail" from the summit to the bottom. The "steepest trail" starts at the summit of the slope and proceeds down from one row to the next, ending at a location in the bottom row. On each step down, the trail proceeds to one of the two or three neighboring locations in the row below it: just below the current location or the one diagonally to the left or the one diagonally to the right, staying within the bounds of the `alts` array. Of the available two or three options, the method always chooses the location with the lowest altitude. If two or all three neighboring locations below the current one have the same smallest altitude, then any one of them may be chosen. The figure below shows an example of the altitude values on a slope and the locations that make up the "steepest trail."



The method `findSteepestTrail()` for the slope returns an `ArrayList` of `Integer` objects whose values are the indices of the altitudes in the respective rows, starting with the location of the summit and ending with a location in the bottom row of the slope. In the above example, the indices of the circled locations are 1, 0, 1, 2, 3, 3, 2. Assume that the `findSummit` method from Part (a) works as specified, regardless of what you wrote there, and do not duplicate its code.

Complete the method `findSteepestTrail` below.

```
/** Builds and returns the "steepest trail" on the slope.
 *  The trail starts at the summit, in the top row. On each
 *  step, the trail proceeds to the neighboring location
 *  just below the current location or the one diagonally
 *  to the left or to the right (if available), always
 *  choosing the location with the lowest altitude. (If the
 *  altitude in two or three neighboring locations is the
 *  same, any one of them may be chosen.) Returns an
 *  ArrayList<Integer> that holds the indices of locations
 *  on the trail in consecutive rows, starting at the summit.
 */
public ArrayList<Integer> findSteepestTrail()
```

For additional practice, write a method `findZigzagTrail` that builds a trail that starts at the summit and proceeds down from one row to the next, always choosing the location with the greatest altitude in the row below it. Alternate between choosing the leftmost and the rightmost maximum location in consecutive rows: leftmost maximum in the top row, rightmost maximum in the next row, and so on. Write a helper method `leftAndRightMax` that returns a two-element array that holds the indices of the left maximum and the right maximum in a given array.