University Interscholastic League

Computer Science Competition

Number 126 (Invitational B - 2011)

General Directions (Please read carefully!):

1) DO NOT OPEN EXAM UNTIL TOLD TO DO SO.

2) **NO CALCULATOR OF ANY KIND MAY BE USED.**

3) There are 40 questions on this contest exam. You have 45 minutes to complete this contest. If you are in the process of actually writing an answer when the signal to stop is given, you may finish writing that answer.

4) Papers may not be turned in until 45 minutes have elapsed. If you finish the test before the end of the allotted time, remain at your seat and retain your paper until told to do otherwise. Use this time to check your answers.

5) All answers must be written on the answer sheet/Scantron card provided. Indicate your answers in the appropriate blanks provided on the answer sheet or on the Scantron card. Clean erasures are necessary for accurate Scantron grading.

6) You may place as many notations as you desire anywhere on the test paper, but not on the answer sheet or Scantron card which are reserved for answers only.

7) You may use additional scratch paper provided by the contest director.

8) All questions have ONE and only ONE correct (BEST) answer. There is a penalty for all incorrect answers. **All provided code segments are intended to be syntactically correct, unless otherwise stated. Ignore any typographical errors and assume any undefined variables are defined as used.**

9) A reference to commonly used Java classes is provided at the end of the test, and you may use this reference sheet during the contest. You may detach the reference sheets from the test booklet, but DO NOT DO SO UNTIL THE CONTEST BEGINS.

10) Assume that any necessary import statements for standard Java packages and classes (e.g. `.util`, `ArrayList`, etc.) are included in any programs or code segments that refer to methods from these classes and packages.

Scoring:

1) All questions will receive **6 points** if answered correctly; no points will be given or subtracted if unanswered; **2 points** will be deducted for an incorrect answer.

QUESTION 1

What does 315_{16} minus $A27_{16}$ equal?

A. $-8EE_{16}$ B. $-6EE_{16}$ C. $-D3C_{16}$ D. -712_{16} E. -1810_{16}

QUESTION 2

What is output by the code to the right?

A. `60` B. `1000` C. `0`

D. `30` E. `20`

```
int x = 5;
int y = 2;
int z = (x * y + y * x) % 1000;
System.out.print(z);
```

QUESTION 3

What is output by the code to the right?

A. `1` B. `11` C. `12`

D. `15` E. `24`

```
int total = 0;
for(int i = -2; i <= 12; i++)
  total++;
System.out.print(total);
```

QUESTION 4

What is output by the code to the right?

A. `gel` B. `elb`

C. `elba` D. `gelb`

E. `Engelbart`

```
String res = "Engelbart".substring(2, 5);
System.out.print(res);
```

QUESTION 5

What is output by the code to the right?

A. `0` B. `1`

C. `false` D. `true`

E. The output will vary from one run of the program to the next.

```
boolean[] flags = new boolean[5];
System.out.print(flags[2]);
```

QUESTION 6

What is output by the code to the right?

A. `25 20` B. `1 80` C. `25 80`

D. `5 80` E. `5 20`

```
int x2 = 105;
int y2 = 20;
x2 %= y2 * 4;
System.out.print(x2 + " " + y2);
```

QUESTION 7

How many combinations of values for the `boolean` variables `p`, `q`, and `r` will result in `s` being set to `true`?

A. `8` B. `5` C. `4`

D. `3` E. `1`

```
boolean p, q, r;
//code to initialize p, q, and r

boolean s = p || (q && r);
```

QUESTION 8

What is output by the code to the right?

A. `1` B. `2` C. `3`

D. `12` E. `13`

```
String s1 = "25A";
if(s1 != null && s1.length() > 5)
  System.out.print(1);
if(s1.contains("5"))
  System.out.print(2);
else
  System.out.print(3);
```

QUESTION 9

What is output by the statement in client code to the right marked `// line 1`?

A. `0`

B. `1`

C. `2`

D. `3`

E. The output will vary from one run of the program to the next.

QUESTION 10

What is output by the statement in client code to the right marked `// line 2`?

A. `0`

B. `1`

C. `2`

D. `3`

E. The output will vary from one run of the program to the next.

```
public class Critter{
  public static final int NORTH = 0;
  public static final int EAST = 1;
  public static final int SOUTH = 2;
  public static final int WEST = 3;

  private int dir;

  public int move(){
    dir = (dir + 1) % 4;
    return dir;
  }
}

public class Bear extends Critter{
  public int move(){
    return Critter.WEST;
  }
}

// client code
Critter c1 = new Critter();
Critter c2 = new Bear();
for(int i = 0; i < 3; i++){
  c1.move();
  c2.move();
}
System.out.print(c1.move()); // line 1
System.out.print(c2.move()); // line 2
```

QUESTION 11

What is output by the code to the right?

A. `1` B. `125` C. `1003`

D. `8000` E. `1000000`

```
int m = 1000;
m = m >> 3;
System.out.print(m);
```

QUESTION 12

What is output by the code to the right?

A. `4.0` B. `4.2` C. `4.26`

D. `5.0` E. `18.0`

```
double v2 = Math.floor(Math.sqrt(18));
System.out.print(v2);
```

Question 13

What is output by the code to the right when method `pri` is called?

A. `10p-10p` B. `p10-p10`

C. `p10-10p` D. `pp10-101`

E. `pp10-10`

```
public int p2(int x){
  System.out.print("p");
  return x * x + 1;
}

public void pri(){
  int y = 3;
  System.out.print( p2(y) + "-" + p2(y) );
}
```

Question 14

What is output by the code to the right? ƀ indicates a space.

A. `732`

B. ƀƀƀ`732`

C. `%732`

D. `732.000`

E. `000732`

```
System.out.printf("%06d", 732);
```

Question 15

What is returned by the method call `exec(5)`?

A. `0` B. `1` C. `15`

D. `32` E. `120`

```
public int exec(int n){
  if(n == 0)
    return 0;
  else
    return n + exec(n - 1);
}
```

Question 16

What is output by the code to the right?

A. `15` B. `18` C. `30`

D. `45` E. `125`

```
String sts = "";
for(int i = 0; i < 5; i++){
  for(int j = 0; j < 3; j++)
    sts += "*";
  for(int j = 0; j < 3; j++)
    sts += "*";
}
System.out.print(sts.length());
```

Question 17

What is output by the client code to the right?

A. `0`

B. `1`

C. `Rating class`

D. There is no output due to a syntax error in the `Rating` class.

E. There is no output due to a runtime error.

```
public class Rating {

  private int numStars;

  public void show(){
    System.out.print(numStars);
  }

  public static void show(){
    System.out.print("Rating class");
  }
}

// client code
Rating r = new Rating();
r.show();
```

QUESTION 18

What replaces **<*1>** in the code to the right so that the output is `4`?

A. `\\s+` B. `_*&` C. `[_*&]`

D. `\\S+` E. `\\_\\*\\&`

```
String names = "Ted_Kenny*Ben&Ray";
String[] info = names.split("<*1>");
System.out.print(info.length);
```

QUESTION 19

Which of the following replaces **<*1>** in the code to the right to indicate `Piece` is a data type that cannot be instantiated and so that the `Piece` class will compile without error.

I. `abstract`
II. `abstract class`
III. `interface`

A. I only B. II only C. III only

D. I and II only E. I, II, and III

```
public <*1> Piece {

  private String name;

  public Piece(String n) {
    name = n;
  }

  public String toString(){
    return name;
  }
}
```

QUESTION 20

What is the smallest possible value that will be printed when method `alpha` is called?

A. `-2147483648` B. `-1` C. `0`

D. `1` E. `2`

```
public void alpha(int x){
  int y = 1;
  do
    y *= 2;
  while( y < x );
  System.out.print(y);
}
```

QUESTION 21

What is output by the code to the right?

A. `[D, F, C]` B. `[D, B, C, F]`

C. `[D, B, CF]` D. `[B, D, F]`

E. `[D, B, F]`

```
ArrayList<String> abr;
abr = new ArrayList<String>();
abr.add("B");
abr.add("C");
abr.add(1, "D");
abr.set(2, "F");
System.out.print(abr);
```

QUESTION 22

What is output by the code to the right?

A. `false false` B. `false true`

C. `true false` D. `true true`

E. There is no output due to a syntax error.

```
List<Integer> sc;
sc = new LinkedList<Integer>();
boolean b1 = sc instanceof Collection;
boolean b2 = sc instanceof ArrayList;
System.out.print(b1 + " " + b2);
```

QUESTION 23

Which of the following can replace **<*1>** in the code to the right so that the code segment compiles without error?

A. `recs.iterator`

B. `new Iterator`

C. `new Iterator<Double>`

D. `recs.iterator<Double>`

E. `new Iterator<double>`

```
ArrayList<Double> recs;
recs = new ArrayList<Double>();
recs.add(12.4);
recs.add(15.3);
Iterator<Double> it = <*1>();
```

QUESTION 24

What is output by the client code to the right?

A. `-1` B. `0` C. `2`

D. `3` E. `5`

QUESTION 25

Which search algorithm does method `s2` implement?

A. heap search B. binary search

C. hash search D. radix search

E. sequential search

```
public int s2(int[] v, int t, int p) {
  if(p == v.length)
    return -1;
  else if(v[p] == t)
    return p;
  return s2(v, t, p + 1);
}

public int s1(int[] v, int t) {
  return s2(v, t, 0);
}

// client code
int[] figs = {-5, 2, 3, -1, 2, -1, 12};
System.out.print(s1(figs, -1));
```

QUESTION 26

Which of the following is not a Java keyword?

A. `super` B. `continue` C. `this` D. `case` E. `List`

QUESTION 27

What replaces **<*1>** so that method `sort` compiles without error and always sorts the values in `list` into ascending order?

A. `list[j - 1] = t`

B. `list[j--] = t`

C. `list[--j] = t`

D. `list[j] = t`

E. `list[i - 1] = t`

Assume **<*1>** is filled in correctly.

QUESTION 28

What sorting algorithm does method `sort` implement?

A. radix sort B. insertion sort

C. selection sort D. merge sort

E. heap sort

```
public void sort(int[] list) {
  int t, j;
  for(int i = 1; i < list.length; i++) {
    t = list[i];
    j = i;
    while((j > 0) && (t < list[j - 1])) {
      list[j] = list[j - 1];
      <*1>;
    }
  }
}
```

QUESTION 29

What is output by the code to the right when method `mu` is called?

A. `2` B. `6` C. `9`

D. `12` E. `36`

```
public int theta(int x) {
  int y = 3 * x;
  x *= 3;
  return x + y;
}

public void mu(){
  int y = 2;
  System.out.print(theta(y));
}
```

QUESTION 30

What is output by the code to the right?

A. `15.75` B. `8` C. `15`

D. There is no output due to a syntax error.

E. There is no output due to a runtime error.

```
double a = -3.75;
int x = 12;
x -= a;
System.out.print(x);
```

QUESTION 31

Given the undirected, weighted graph to the right, what is the cost of the lowest cost path from vertex A to vertex E?

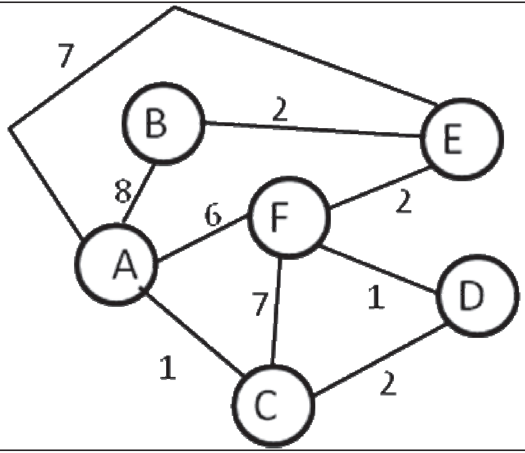
A. `6`

B. `7`

C. `8`

D. `9`

E. `10`



QUESTION 32

What is the Big O of method `fill` if the preconditions of the method are met and `c` is the following type of `Collection`? N = `n`. Pick the most restrictive correct set of answers.

	`ArrayList<Integer>`	`TreeSet<Integer>`
A.	$O(N^2)$	$O(N^2)$
B.	$O(N^2)$	O(N)
C.	O(N)	O(N)
D.	O(NlogN)	$O(N^2)$
E.	O(N)	O(NlogN)

```
// pre: c != null, c.size() == 0
public void fill(Collection<Integer> c,
                                    int n) {
  for(int i = 0; i < n; i++)
    c.add(i);
}
```

QUESTION 33

What is output by the code to the right?

A. `22`

B. `11`

C. `9`

D. `8`

E. `6`

```
Set<Character> s1, s2;
s1 = new TreeSet<Character>();
s2 = new HashSet<Character>();
String n1 = "ORCAARRCAAC";
String n2 = "RRECEETETRE";
for(int i = 0; i < n1.length(); i++) {
  s1.add(n2.charAt(i));
  s2.add(n1.charAt(i));
}
s1.addAll(s2);
System.out.print(s1.size());
```

QUESTION 34

Given method `strange` to the right what is output by the following client code?

```
int[] vals = {2, 1, 3, 4, 1, 2, 1};
System.out.println(strange(vals));
```

A. `0` B. `1` C. `2`

D. `10` E. `14`

QUESTION 35

Given method `strange` to the right what is output by the following client code?

```
int[] vals2 = {-2, 1, -3, 4, -1, 2, 1,
                                    -5, 4};
System.out.println(strange(vals2));
```

A. `12` B. `-11` C. `1`

D. `4` E. `6`

```
public int strange(int[] list) {
  int m = 0;
  int h = 0;
  for(int val : list) {
    h = Math.max(0, h + val);
    m = Math.max(h, m);
  }
  return m;
}
```

QUESTION 36

What is output by the code to the right?

A. `10` B. `1`

C. `0` D. `-1`

E. The output will vary from one run of the program to the next.

```
ArrayList<String> titles;
titles = new ArrayList<String>();
System.out.print(titles.size());
```

QUESTION 37

Which of the following best explains why the `ArIt` class to the right will not compile?

A. Classes that implement the `Iterator` interface cannot be declared `public`.

B. The constructor header must be changed from `public ArIt(ArrayList<E> a)` to `public ArIt<E>(ArrayList<E> a)`

C. The instance variable `ar` must be `public`.

D. The keyword `implements` must be changed to `extends`.

E. The `ArIt` class does not implement the `remove` method as specified in the `Iterator` interface.

```
public class ArIt<E> implements
                              Iterator<E>{

  private ArrayList<E> ar;
  private int pos;

  public ArIt(ArrayList<E> a) { ar = a; }

  public boolean hasNext(){
    return pos < ar.size();
  }

  public E next(){ return ar.get(pos++); }
}
```

QUESTION 38

The following values are inserted one at a time in the order shown (left to right) into a binary search tree using the traditional insertion algorithm. What is the height of the resulting tree? The height of a tree is the number of links from the root node to the deepest leaf.

`-5, 6, 15, 8, 17, 32, 8, 9, 10`

A. 5 B. 4 C. 3 D. 2 E. 1

QUESTION 39

Which of the following can replace **`<*1>`** in the code to the right so that method `ct` compiles without error?

I. `Exception bad`
II. `IOException e`
III. `Exception exec`

A. I only B. II only

C. III only D. I and III only

E. I, II, and III

```
public static int ct(String s) {
  int x = 1;
  try{
    Scanner sc = new Scanner(new File(s));
    while(sc.hasNext()) {
      x++;
      sc.next();
    }
  }
  catch(<*1>) { x = 2; }
  finally { x *= -2; }
  return x;
}
```

Assume **`<*1>`** is filled in correctly.

QUESTION 40

What is returned by the method call `ct("X.txt")` if the file `X.txt` cannot be found?

A. -1 B. -2

C. -4 D. 1

E. `FileNotFoundException`

No Test Material on This Page

Standard Classes and Interfaces — Supplemental Reference

`class java.lang.Object`
- `boolean equals(Object other)`
- `String toString()`
- `int hashCode()`

`interface java.lang.Comparable<T>`
- `int compareTo(T other)`
  Return value < 0 if `this` is less than `other`.
  Return value = 0 if `this` is equal to `other`.
  Return value > 0 if `this` is greater than `other`.

`class java.lang.Integer implements Comparable<Integer>`
- `Integer(int value)`
- `int intValue()`
- `boolean equals(Object obj)`
- `String toString()`
- `int compareTo(Integer anotherInteger)`
- `static int parseInt(String s)`

`class java.lang.Double implements Comparable<Double>`
- `Double(double value)`
- `double doubleValue()`
- `boolean equals(Object obj)`
- `String toString()`
- `int compareTo(Double anotherDouble)`
- `static double parseDouble(String s)`

`class java.lang.String implements Comparable<String>`
- `int compareTo(String anotherString)`
- `boolean equals(Object obj)`
- `int length()`
- `String substring(int begin, int end)`
  Returns the substring starting at index `begin` and ending at index `(end - 1)`.
- `String substring(int begin)`
  Returns `substring(from, length()).`
- `int indexOf(String str)`
  Returns the index within this string of the first occurrence of `str`. Returns `-1` if `str` is not found.
- `int indexOf(String str, int fromIndex)`
  Returns the index within this string of the first occurrence of `str`, starting the search at the specified index.. Returns `-1` if `str` is not found.
- `charAt(int index)`
- `int indexOf(int ch)`
- `int indexOf(int ch, int fromIndex)`
- `String toLowerCase()`
- `String toUpperCase()`
- `String[] split(String regex)`
- `boolean matches(String regex)`

`class java.lang.Character`
- `static boolean isDigit(char ch)`
- `static boolean isLetter(char ch)`
- `static boolean isLetterOrDigit(char ch)`
- `static boolean isLowerCase(char ch)`
- `static boolean isUpperCase(char ch)`
- `static char toUpperCase(char ch)`
- `static char toLowerCase(char ch)`

`class java.lang.Math`
- `static int abs(int a)`
- `static double abs(double a)`
- `static double pow(double base, double exponent)`
- `static double sqrt(double a)`
- `static double ceil(double a)`
- `static double floor(double a)`
- `static double min(double a, double b)`
- `static double max(double a, double b)`
- `static int min(int a, in b)`
- `static int max(int a, int b)`
- `static long round(double a)`
- `static double random()`
  Returns a `double` value with a positive sign, greater than or equal to `0.0` and less than `1.0`.

`interface java.util.List<E>`
- `boolean add(E e)`
- `int size()`
- `Iterator<E> iterator()`
- `ListIterator<E> listIterator()`
- `E get(int index)`
- `E set(int index, E e)`
  Replaces the element at `index` with the object `e`.
- `void add(int index, E e)`
  Inserts the object `e` at position `index`, sliding elements at position `index` and higher to the right (adds 1 to their indices) and adjusts `size`.
- `E remove(int index)`
  Removes element from position `index`, sliding elements at position `(index + 1)` and higher to the left (subtracts 1 from their indices) and adjusts `size`.

`class java.util.ArrayList<E> implements List<E>`

`class java.util.LinkedList<E> implements List<E>, Queue<E>`

Methods in addition to the `List` methods:
- `void addFirst(E e)`
- `void addLast(E e)`
- `E getFirst()`
- `E getLast()`
- `E removeFirst()`
- `E removeLast()`

class java.util.Stack<E>
- `boolean isEmpty()`
- `E peek()`
- `E pop()`
- `E push(E item)`

interface java.util.Queue<E>
- `boolean add(E e)`
- `boolean isEmpty()`
- `E peek()`
- `E remove()`

class java.util.PriorityQueue<E>
- `boolean add(E e)`
- `boolean isEmpty()`
- `E peek()`
- `E remove()`

interface java.util.Set<E>
- `boolean add(E e)`
- `boolean contains(Object obj)`
- `boolean remove(Object obj)`
- `int size()`
- `Iterator<E> iterator()`
- `boolean addAll(Collection<? extends E> c)`
- `boolean removeAll(Collection<?> c)`
- `boolean retainAll(Collection<?> c)`

class java.util.HashSet<E> implements Set<E>

class java.util.TreeSet<E> implements Set<E>

interface java.util.Map<K,V>
- `Object put(K key, V value)`
- `V get(Object key)`
- `boolean containsKey(Object key)`
- `int size()`
- `Set<K> keySet()`
- `Set<Map.Entry<K, V>> entrySet()`

class java.util.HashMap<K,V> implements Map<K,V>

class java.util.TreeMap<K,V> implements Map<K,V>

interface java.util.Map.Entry<K,V>
- `K getKey()`
- `V getValue()`
- `V setValue(V value)`

interface java.util.Iterator<E>
- `boolean hasNext()`
- `E next()`
- `void remove()`

interface java.util.ListIterator<E> extends java.util.Iterator<E>

Methods in addition to the `Iterator` methods:
- `void add(E e)`
- `void set(E e)`

class java.lang.Exception
- `Exception()`
- `Exception(String message)`

class java.util.Scanner
- `Scanner(InputStream source)`
- `boolean hasNext()`
- `boolean hasNextInt()`
- `boolean hasNextDouble()`
- `String next()`
- `int nextInt()`
- `double nextDouble()`
- `String nextLine()`
- `Scanner useDelimiter(String pattern)`