Cypress Woods High School

Computer Science Competition

December 2013

General Directions (Please read carefully!):

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

2) **NO CALCULATORS 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. You may 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.

11) Correct responses are based on Java JDK v7.0, from Sun Microsystems, Inc.

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 is the sum of 212_3 and 84_9?

A. 10120_3 B. 11001_3 C. 123_9 D. 120_9 E. 106_{10}

QUESTION 2

What is output by the code to the right?

A. 38 B. 47 C. 39

D. 46 E. 0

```
int a = 23;
int b = a % 2 + a * 2;
System.out.print( b );
```

QUESTION 3

What is output by the code to the right?

A. 4 B. 5 C. 6

D. 7 E. 8

```
int sum = -1;
for(int i = 1; i < 212; i*=2)
  sum++;
System.out.print( sum );
```

QUESTION 4

What is output by the code to the right?

A. PositiveAffirmation!!

B. Pos!tiveAffirmation!!

C. Pos!t!veAff!rmat!onii

D. Pos!t!veAff!rmat!on!!

E. Aff!rmat!on!!Pos!t!ve

```
String s = "Positive";
s += "Affirmation!!";
System.out.println( s.replace('i', '!') );
```

QUESTION 5

What is output by the code to the right?

A. 1 B. 2

C. 3 D. 5

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

```
int[] ar = {1, 2, 3, 4, 5};
System.out.print( ar[ar.length-1 / 2] );
```

QUESTION 6

What is output by the code to the right?

A. 132 B. 126

C. 26 D. 32

E. 0

```
int a = 42;
a *= 3;
System.out.println( a % 100 );
```

QUESTION 7

What is output by the code to the right?

A. `true` B. `false` C. `null`

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

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

```
Boolean b;
b = true && b;
System.out.print( b );
```

QUESTION 8

What is output by the code to the right?

A. `n9` B. `n10` C. `n11`

D. `o9` E. `o10`

```
int x = 10;
if( x < 9 || x + 1 > 10 )
  System.out.print( "n" + x );
else
  System.out.print( "o" + x );
```

QUESTION 9

What replaces **<*1>** in the code to the right so that the class `Tree` compiles without error?

A. `package` B. `class` C. `extends`

D. `abstract` E. `implements`

```
public class Tree <*1> Object{

  private Tree(){
  }

  public static double log(double x){
    return x * 10;
  }
}

///////// client code /////////
System.out.println(Tree.log(10));
```

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

QUESTION 10

What is the output of the client code?

A. `1.0`

B. `100.0`

C. `10`

D. `100`

E. `10.0`

QUESTION 11

What is output by the code to the right?

A. `0` B. `8` C. `18`

D. `2` E. `12`

```
byte x = 10;
byte y = 8;
System.out.print( x ^ y );
```

QUESTION 12

What is output by the code to the right?

A. `-5.0 -5` B. `-5.0 -6` C. `-6.0 -5`

D. `-6.0 -6` E. `-5.5 -6`

```
System.out.print( Math.ceil( -5.5 ) + " ");
System.out.print( Math.round( -5.5 ) );
```

QUESTION 13

What is output by the code to the right?

A.
```
http:\google.com
ews'a'
```

B.
```
http:\\google.com\news\'a'
```

C.
```
http:\google.com
news'a'
```

D.
```
http:\google.com\news'a'
```

E.
```
http:\google.com
news\'a'
```

```
System.out.print("http:\\google");
System.out.print(".com\news");
System.out.println("\'a'");
```

QUESTION 14

What is output by the code to the right?

A. `3.1 p` B. `3.1 pi`

C. `3.14 pi` D. `33.14 1pi`

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

```
System.out.printf("%3s %1s", 3.14, "pi");
```

QUESTION 15

What is returned by the method call `abc(2,5)`?

A. `50` B. `10` C. `5`

D. `70` E. `7`

```
public static int abc(int x, int y){
  x = y;
  y = x;
  x += y;
  return y * x;
}
```

QUESTION 16

Which of the following replaces **<*1>** in the code to the right so that it compiles without error?

A. `10`

B. `new int[10]`

C. `new int[10][10]`

D. `new int[][10]`

E. `new[] int[10]`

```
int[] ar[] = <*1>;
```

QUESTION 17

What is output by the code to the right?

A. `7` B. `11`

C. `8` D. `10`

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

```
Object[] ar = {true, 3.00, 1024, 'a'};
String s = ar[1].toString();
s += ar[2].toString();
System.out.print( s.length() );
```

QUESTION 18

What is output by the code to the right?

A. `01234` B. `014916`

C. `14916` D. `1234`

E. `01496`

```
for(int i = 0; i<=4; i++)
   System.out.print(i * i % 10);
```

QUESTION 19

What is output by the code to the right?

A. `3.1 3` B. `2.7 3` C. `2.1 2`

D. `3.1 2` E. `3.1 2.0`

```
double p = 3.14;
double q = 2.12;
System.out.printf( "%.1f %.0f", q,q );
```

QUESTION 20

Which of the following statements compiles successfully?

I. `byte b1 = (int)(65 + 3);`

II. `Byte b2 = (int)(65 + 3);`

III. `Integer i3 = (byte)(65 + 3);`

A. I only B. III only C. I and II only D. I and III only E. None of the above

QUESTION 21

What is output by the code to the right?

A. `0` B. `3` C. `1`

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

E. There is no output due to a `StackOverflowError`.

```
Stack<Integer> overflow;
overflow = new Stack<Integer>();
overflow.push(3);
System.out.println(overflow.get(0));
```

QUESTION 22

What is returned by the method call `recur(16,12)`?

A. `48` B. `16` C. `12`

D. `4` E. `1`

QUESTION 23

What is the purpose of the `recur` method?

A. Returns the maximum of `a` and `b`

B. Returns the minimum of `a` and `b`

C. Returns the greatest common factor of `a` and `b`

D. Returns the least common multiple of `a` and `b`

E. Returns the largest prime factor of `a` and `b`

```
public static int recur(int a, int b){
  if( a == 0 ) return b;
  return recur( b%a, a );
}
```

QUESTION 24

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

A. `[1, 3, 4, 5, 9]`

B. `[1, 1, 1, 1, 1]`

C. `[3, 4, 1, 5, 9]`

D. `[9, 5, 4, 3, 1]`

E. `[4, 3, 9, 5, 1]`

QUESTION 25

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

A. `[9, 5, 5, 4, 4, 3, 1]`

B. `[1, 1, 2, 2, 2, 2, 1]`

C. `[1, 3, 4, 5, 9, 5, 9]`

D. `[1, 3, 4, 4, 5, 5, 9]`

E. `[3, 4, 5, 1, 4, 5, 9]`

QUESTION 26

What is the worst case Big O runtime of `magic(ar)` when n = `ar.length`? Choose the most restrictive, correct answer.

A. $O(\log_2 n)$

B. $O(n \log_2 n)$

C. $O(n)$

D. $O(n^2)$

E. $O(n^2 \log_2 n)$

QUESTION 27

What algorithm does the method `magic(int[] ar)` implement?

A. Insertion sort

B. Quick sort

C. Merge sort

D. Binary search

E. None of the above

```
// pre: all values in ar are < 100 and >= 0
public static void magic(int[] ar){
  int[] c = new int[100];
  for(int i : ar)
    c[i]++;
  int i = 0, j = ar.length-1;
  for(int k : c){
    while(k-->0)
      ar[j--] = i;
    i++;
  }
}

///////// client code /////////
int[] a1 = {4,3,5,1,9};
magic(a1);
String s1 = Arrays.toString(a1);
System.out.println(s1); //line 1

int[] a2 = {4,3,5,1,4,5,9};
magic (a2);
String s2 = Arrays.toString(a2);
System.out.println(s2); //line 2
```

QUESTION 28

What replaces **<*1>** in the code to the right to obtain the character at position `i` in the `String s`?

A. `s[i]`

B. `s.char(i)`

C. `s.substring(i)`

D. `s.substring(i, i+1)`

E. `s.charAt(i)`

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

QUESTION 29

What replaces **<*2>** in the code to the right to check if `ch` is a letter?

A. `Character.isLetter(ch)`

B. `Character.letter(ch)`

C. `ch.isLetter()`

D. `ch.letter`

E. `char.isLetter(ch)`

Assume **<*2>** is filled in correctly.

QUESTION 30

What is returned by the method call `test("ac3d90")`?

A. `0`

B. `3`

C. `true`

D. `false`

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

QUESTION 31

What is returned by the method call `test("b31xy7")`?

A. `3`

B. `6`

C. `true`

D. `false`

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

```
public static boolean test(String s){
  int ct = 0;
  for(int i = 0; i < s.length(); i++){
    char ch = <*1>;
    if( <*2> )
      ct++;
    else
      ct--;
    if( ct < 0 )
      return false;
  }
  return ct == 0;
}
```

QUESTION 32

What is output by the code to the right?

A. `4777` B. `4de` C. `122`

D. `3777` E. `123`

```
String s = "123ab456de777fg";
String[] sr = s.split("\\D*");
System.out.println(sr.length + sr[2]);
```

QUESTION 33

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

A. `a` B. `e` C. `S`

D. `G` E. a space character

QUESTION 34

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

A. `e` B. `s` C. `a`

D. `G` E. `t`

```
PriorityQueue<Character> pq;
pq = new PriorityQueue<Character>();

String s = "Steve Gates";
for(char c : s.toCharArray())
  pq.add(c);

pq.remove();
pq.remove();
System.out.println(pq.remove()); //line 1
pq.remove();
System.out.println(pq.remove()); //line 2
```

QUESTION 35

How many neighbors does node 2 have in the graph to the right?

A. 2

B. 5

C. 6

D. 114

E. 8

QUESTION 36

What kind of graph is displayed on the right?

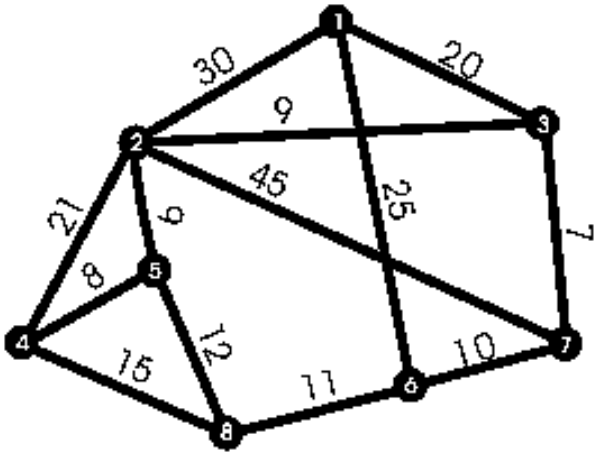
A. A undirected, unweighted graph

B. A undirected, weighted graph

C. A directed, unweighted graph

D. A directed, weighted graph

E. A binary tree



QUESTION 37

What replaces **<*1>** in the code to the right to correctly instantiate `t` to an array of size `n`?

A. `new Array(n)`

B. `new Array<int>(n)`

C. `new int(n)`

D. `new int[n]`

E. `new int[](n)`

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

QUESTION 38

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

A. `0`

B. `2`

C. `7`

D. `4`

E. `3`

QUESTION 39

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

A. `4`

B. `6`

C. `5`

D. `3`

E. `9`

QUESTION 40

What Java language feature does the method `get` employ?

A. Autoboxing

B. Multiple inheritance

C. Method overriding

D. Polymorphism

E. Method overloading

```
class Structure{
  private int[] t;
  private int n;
  public Structure(int size){
     n = size;
     t = <*1>;
  }
  public void add(int x){
     for(; x<n; x+=x&-x)
        t[x]++;
  }
  public int get(int x){
     int s = 0;
     for(; x>0; x-=x&-x)
        s += t[x];
     return s;
  }
  public int get(int a, int b){
     return get(b)-get(a-1);
  }
  public int size(){
     return n;
  }
}


///////// client code /////////
Structure s = new Structure(10);
s.add(3);
s.add(4);
s.add(8);
System.out.println(s.get(7)); //line 1

s.add(9);
s.add(8);
int x = s.get(4,9);
System.out.println(x); //line 2
```

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()`

class java.util.ArrayList<E> implements List<E>
Methods in addition to the `List` methods:
- `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.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)