University Interscholastic League

Computer Science Competition

UTCS UIL Open - 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 is the sum of 713_{16} and $79A_{16}$?

A. EAD_{16} B. $F00_{16}$ C. $F04_{16}$ D. $FB4_{16}$ E. $EB4_{16}$

QUESTION 2

What is output by the code to the right?

A. `6 15` B. `5 15` C. `5 18`

D. `7 15` E. `6 18`

```
int x = 5;
int y = 3 * x;
x++;
System.out.print(x + " " + y);
```

QUESTION 3

What is output by the code to the right?

A. `5 5` B. `10` C. `6 5`

D. `6` E. `11`

```
int total = 0;
int i = 1;
for(;i <= 5; i++)
  ++total;
System.out.print(total + i);
```

QUESTION 4

What is output by the code to the right?

A. `Cal` B. `vinLin` C. `Lin`

D. `lvi` E. `lvinLin`

```
String name = "CalvinLin";
name = name.substring(3);
System.out.println(name);
```

QUESTION 5

What is output by the code to the right?

A. `easyBDF` B. `ABCABC`

C. `easyABCABC` D. `easyABC`

E. `easyABCeasyABC`

```
String lets = "ABC";
System.out.print( "easy" + lets + lets);
```

QUESTION 6

What is output by the code to the right?

A. `7.0` B. `7.125` C. `3.125`

D. `5.125` E. `5.0`

```
double a2 = 35.125;
double b2 = a2 % 10 + 2;
System.out.print(b2);
```

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. `7` C. `3`

D. `1` E. `0`

```
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. `BC` B. `AC` C. `BD`

D. `AD` E. There is no output.

```
int x3 = 7;
if(x3 != 5 || x3 != 7)
  System.out.print("A");
else
  System.out.print("B");
if(x3 != 5 && x3 != 7)
  System.out.print("C");
else
  System.out.print("D");
```

Question 9

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

A. `3`

B. `2`

C. `1`

D. `0`

E. `true`

Question 10

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

A. `0`

B. `3`

C. `b10`

D. `b13`

E. `null0`

```
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 == 0 ? WEST : NORTH;
    return dir;
  }
}

public class Badger extends Critter{
  private String name;

  public Badger(String s){ name = s; }

  public String toString(){
    return name;
  }
}

// client code
Critter c1 = new Critter();
Badger b1 = new Badger("b1");
c1.move();
System.out.print(c1.move()); // line 1

String stc = b1.toString() + b1.move();
System.out.println(stc); // line 2
```

Question 11

What is output by the code to the right?

A. `0` B. `40` C. `125`

D. `625` E. `32768`

```
int m = 5 << 3;
System.out.print(m);
```

Question 12

What is the maximum possible number of `'*'`s the code to the right will print when run?

A. `0` B. `1` C. `7`

D. `8` E. `9`

```
double limit = Math.random() * 8;
for(int i = 0; i <= limit; i++)
  System.out.print('*');
```

QUESTION 13

What is output by the code to the right?

A. `X1X2Y1` B. `"X1\"X2\"Y1"`

C. `X1"X2"Y1` D. `"X1""\"X2\"""Y1"`

E. `X1\"X2\"Y1`

```
System.out.print("X1");
System.out.print("\"X2\"");
System.out.println("Y1");
```

QUESTION 14

What is output by the code to the right?

A. `009.10` B. `9.109382` C. `  9.10`

D. `009.11` E. `9.11`

```
int x = 2;
String f = "%" + (x+x) + "." + x + "f";
System.out.printf(f, 9.109382);
```

QUESTION 15

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

A. `17` B. `11.5` C. `11`

D. `9.5` E. `9`

```
public int manip(int x){
  int y = x + 2;
  x /= 2;
  return y + x;
}
```

QUESTION 16

What is output by the code to the right?

A. `0` B. `10` C. `11`

D. `18` E. `20`

```
String stars = "";
for(int i = 0; i < 10; i++)
  stars += "*";
for(int i = 0; i < 10; i++)
  stars += "*";
System.out.print(stars.length());
```

QUESTION 17

What replaces **`<*1>`** in the code to the right to indicate method `start` does not return a value?

A. `static` B. `class` C. `final`

D. `void` E. `null`

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

QUESTION 18

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

A. `85` B. `135`

C. `256` D. `266`

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

```
public int other(int x, int y) {
  x--;
  y++;
  return x * y;
}

public int other(int x) {
  x++;
  int y = other(x, x);
  x++;
  return x * y;
}

public <*1> start() {
  System.out.print(other(3) + other(3,4));
}
```

QUESTION 19

What is output by the code to the right?

A. `TexasOrange`

B. `Texas7911497110103101`

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

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

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

```
String st5 = "Texas";
Object ob5 = "Orange";
System.out.print(st5.toString());
System.out.print(ob5.toString());
```

QUESTION 20

What is output by the code to the right?

A. `BC` B. `12A`

C. `C15` D. `14C`

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

```
String sd;
sd = "12 A 13 B 14 C 15";
Scanner sc2 = new Scanner(sd);
for(int i = 0; i < 4; i++)
  sc2.next();
System.out.print(sc2.next());
System.out.print(sc2.next());
```

QUESTION 21

What is output by the code to the right?

A. `452100` B. `4521` C. `452`

D. `4521.00` E. `21`

```
int qq = 4521;
int zz = qq % 100;
System.out.print(zz);
```

QUESTION 22

What is output by the code to the right?

A. `23` B. `19` C. `8`

D. `4` E. `-1`

```
String s = "ABAaaBBAAbAAbAAaBAabbaa";
int val = s.toLowerCase().indexOf("abbaa");
System.out.print(val);
```

QUESTION 23

What is output by the code to the right?

A. `[.3, .2, .1]` B. `[0.3, 0.1, 0.2]`

C. `[0.3, 0.1]` D. `[0.1, 0.2, 0.3]`

E. `[0.1, 0.3]`

```
ArrayList<Double> ds;
ds = new ArrayList<Double>();
ds.add(.1);
ds.add(.2);
ds.set(1, .3);
System.out.print(ds);
```

QUESTION 24

What is output by the code to the right?

A. `afzAA` B. `AAafz` C. `afzA`

D. `afz` E. `Aafz`

```
char[] lets = {'z', 'A', 'f', 'a', 'A'};
Arrays.sort(lets);
String res = "";
for(char ch : lets)
  res += ch;
System.out.print(res);
```

QUESTION 25

What is output by the code to the right?

A. `"ceboo"` B. `ceboo` C. `cebook`

D. `cebo` E. `ebo`

```
String tc = "Facebook";
String part = tc.substring(2, 6);
System.out.print(part);
```

QUESTION 26

What replaces **<*1>** in the code to the right to initialize the variable `total` to zero?

A. `null` B. `false` C. `pts`

D. `0` E. More than one of the answers A through D is correct.

```
int[] pts = {12, 5, 15, 10, 12, 7};
int total = <*1>;
for(int it : pts) {
  int temp = it / 5;
  switch (temp) {
    case 0 : total += 1; break;
    case 1 : total += 2; break;
    case 2 : total += 4; break;
    default : total += 8;
 }
}
System.out.print(total);
```

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

QUESTION 27

What is output by the code to the right?

A. `4` B. `8` C. `11`

D. `24` E. `42`

QUESTION 28

What is output by the code to the right?

A. `[4, 2, 5]` B. `[2, 5, 6]`

C. `[6, 5, 4]` D. `[3, 5, 7]`

E. `[2, 5, 4]`

```
int[] ref1 = {2, 5, 4};
int[] ref2 = {3, 2, 1};
ref2[1]++;
ref2 = ref1;
ref2[2] += ref1[0];
System.out.print(Arrays.toString(ref1));
```

QUESTION 29

What is output by the code to the right?

A. `[0, 1]` B. `[1, 0]`

C. `[12, 13]` D. `[13, 12]`

E. There is no output due to a syntax error in the code.

```
ArrayList<Double> reals;
reals = new ArrayList<Double>();
reals.add(0, 12);
reals.add(1, 13);
System.out.print(reals);
```

QUESTION 30

What is output by the code to the right?

A. `2` B. `8` C. `10`

D. `256` E. `true`

```
int val2 = 2;
int val3 = 8;
int val4 = val2 ^ val3;
System.out.println(val4);
```

QUESTION 31

Which sorting algorithm does not compare the elements being sorted to each other if the maximum value of the elements being sorted in already known?

A. Selection sort B. Insertion sort C. Radix sort D. Heap sort E. Quicksort

QUESTION 32

What replaces `<*1>` in the code to the right so that the instance variable `x` can be accessed by code in all classes?

A. `final` B. `private` C. `public`

D. `void` E. `protected`

```
public class Sample {
  <*1> int x;

  public static void bar(int x) {
    foo();
    x++;
  }

  public void foo() {x++;}
}

// client code
Sample st = new Sample();
st.bar(st.x);
System.out.print(st.x);
```

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

QUESTION 33

What is output by the client code to the right?

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

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

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

QUESTION 34

What kind of graph does the picture to the right represent?

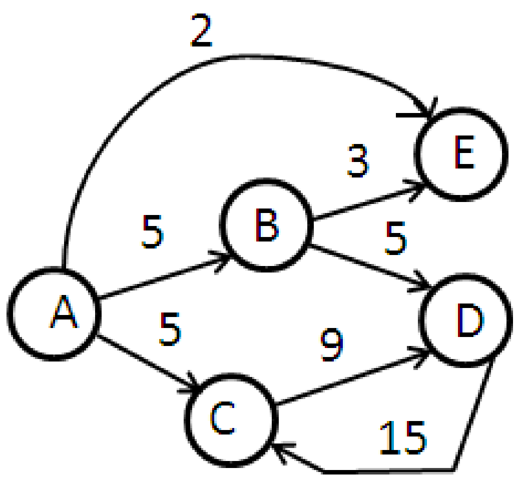
A. a directed unweighted graph

B. a directed weighted graph

C. an undirected unweighted graph

D. an undirected weighted graph

E. a binary search tree



QUESTION 35

What is the cost of the lowest cost path from vertex D to vertex A?

A. 5 B. 10

C. 14 D. 20

E. There is no path from vertex D to vertex A.

Go on to the next page.

QUESTION 36

What replaces **<*1>** and **<*2>** in the code to the right to set the constant `LIM1` to the number of rows in `t` and the constant `LIM2` to the number of columns in `t`?

	<*1>	<*2>
A.	`t.length`	`t.length.length`
B.	`t.length.length`	`t.length`
C.	`t.length`	`t[0].length`
D.	`t[0].length`	`t[1].length`
E.	`t[0].length`	`t.length`

```
// pre: t != null, t refers to a
// rectangular matrix, t.length > 0

public int handle(int[][] t) {
  int total = 0;
  final int LIM1 = <*1>;
  final int LIM2 = <*1>;

  for(int r = 0; r < LIM1; r++) {
    int temp1 = LIM2 / 2;
    int temp2 = 0;
    int temp3 = LIM1 * 2;
    int c = 0;
    while(temp2 < temp3 && c < temp1) {
      temp2 += t[r][c];
      temp2 += t[r][LIM2 - 1 - c++];
    }
    total += temp2;
  }
  return total;
}
```

Assume **<*1>** and **<*2>** are filled in correctly.

QUESTION 37

What is returned by method `handle` if `t` is the 2d array shown below?

1	8	0	9	1	6
0	1	5	4	0	4
2	2	7	1	13	2
11	5	13	13	4	20
0	1	5	4	0	4

A. `44`　　B. `59`

C. `94`　　D. `137`

E. There is no output due to an infinite loop that occurs when the method is called with the 2d array shown above.

Go on to the next page.

QUESTION 38

What is the best case order (Big O) and worst case order of the `insert` method in the `Structure` class to the right given the `Structure` already contains N elements? Pick the most restrictive set of correct answers.

	Best case	Worst Case
A.	O(1)	O(1)
B.	O(1)	O(N)
C.	O(N)	O(N)
D.	O(N)	O(N^2)
E.	O(N^2)	O(N^2)

QUESTION 39

What is output by the following client code?

```
Structure st = new Structure();
System.out.print(st.size());
st.add(4);
st.add(8);
st.add(6);
for(int i = 0; i < st.size(); i++)
  System.out.print(st.get(i));
System.out.print(st.size());
```

A. `104863`

B. `04863`

C. `06843`

D. `10486310`

E. There is no output due to a runtime error caused by the client code.

QUESTION 40

What kind of data structure does the `Structure` class implement?

A. a stack

B. an array based list

C. a queue

D. a linked list

E. a heap

```
public class Structure {
  private Object[] con;
  private int size;

  public Structure() {
    con = new Object[10];
    size = 0;
  }

  public void add(Object x){
    insert(size, x);
  }

  public Object get(int pos){
    return con[pos];
  }

  public void insert(int pos, Object obj){
    ensureCapcity();
    for(int i = size; i > pos; i--)
      con[i] = con[i - 1];
    con[pos] = obj;
    size++;
  }

  public Object remove(int pos){
    Object removedValue = con[pos];
    for(int i = pos; i < size - 1; i++)
      con[i] = con[i + 1];
    con[size - 1] = null;
    size--;
    return removedValue;
  }

  public int size(){
    return size;
  }

  private void ensureCapcity(){
    if(size == con.length)
      resize();
  }

  private void resize() {
    Object[] temp;
    temp = new Object[con.length * 2];
    for(int i = 0; i < con.length; i++)
      temp[i] = con[i];
    con = temp;
  }
}
```

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)`

Computer Science Answer Key
UIL UTCS UIL Open 2011

1. A	11. B	21. E	31. C
2. A	12. D	22. D	32. C
3. E	13. C	23. E	33. D
4. B	14. E	24. B	34. B
5. C	15. E	25. D	35. E
6. B	16. E	26. D	36. C
7. B	17. D	27. D	37. C
8. D	18. A	28. B	38. B
9. D	19. A	29. E	39. B
10. D	20. D	30. C	40. B

Notes:

The clause "Choose the most restrictive correct answer." is necessary because per the formal definition of Big O, an algorithm that is $O(N^2)$ is also $O(N^3)$, $O(N^4)$, and so forth.

29. The compiler will not convert the `int` to a `double` and then autobox the `double` to a `Double` object in this case.

33. `static` methods may not call non `static` methods in the way shown in the class.