

**QUESTION 1**

What is the product of  $573_8$  and  $246_8$ ?

- A.  $140958_8$       B.  $423236_8$       C.  $62914_8$       D.  $172702_8$       E. None of these

**QUESTION 2**

Which of the following replaces **<\*1>** in the code to the right to determine the number of command line arguments passed to `main()`?

- A. `String[].length`  
 B. `String[].length()`  
 C. `args.length`  
 D. `args.lenth()`  
 E. None of these

```
public static void main(String[] args) {
    int numargs = <*1>;
    // more processing not shown
}
```

**QUESTION 3**

Which of the following replaces **<\*1>** in the code to the right to subtract the bitwise OR of `x` and `y` from the bitwise AND of `x` and `y`?

- A. `(x&&y) - (x||y)`  
 B. `x - y`  
 C. `(x&y) - (x|y)`  
 D. `(x&&y) - (x^y)`  
 E. None of these

```
public static int mystery(int x, int y) {
    return <*1>;
}
```

**QUESTION 4**

Assume that **<\*1>** is filled in correctly. What is returned by `mystery(54, 90)`?

- A. 36                      B. 108  
 C. -36                    D. -108  
 E. None of these

**QUESTION 5**

Assume that the section of code marked with **<\*1>** does not modify `i` or `j` and runs in time  $O(n)$ . What is the running time of the code to the right? Choose the smallest correct answer.

- A.  $O(n)$                       B.  $O(n \log n)$   
 C.  $O(n^2)$                     D.  $O(n^3)$   
 E. None of these

```
int n;
// code to initialize n not shown

int i=0, j;
while (++i<n) {
    j=i;
    while (++j<n) {
        <*1>
    }
}
```

**QUESTION 6**

Which of the following replaces each instance of **<\*1>** in the code to the right to declare constant integer values accessible everywhere?

- A. `public int`
- B. `static final int`
- C. `public static final int`
- D. `final public int`
- E. More than one of these

For the remaining questions, assume **<\*1>** has been filled in correctly.

**QUESTION 7**

Which of the following builds a Map named `m` which can be used to map from colleges to mascots?

- A. `Map m = new Map();`
- B. `Map m = new Map(College, Mascot);`
- C. `Map m = new HashMap(Mascot, College);`
- D. `Map m = new TreeMap(College->Mascot);`
- E. None of these

**QUESTION 8**

Assume that Map `m` has been built correctly, and that College `texas` and Mascot `longhorn` have been built correctly. Which of these adds to Map `m` the key `texas` with value `longhorn`?

- A. `m.put(texas.name, longhorn.name);`
- B. `m[texas] = longhorn;`
- C. `m.put(texas->longhorn);`
- D. `m.put(texas, longhorn);`
- E. None of these

**QUESTION 9**

Which of the following can be run outside class `Mascot` to check whether the mascot associated with College `osu` in Map `m` is an animal?

- A. `m.get(osu).isAnimal()`
- B. `m.get(osu).type == Mascot.ANIMAL`
- C. `((Mascot)m.get(osu)).isAnimal()`
- D. `((Mascot)m.get(osu)).type == ANIMAL`
- E. More than one of these

```
public class College {

    // methods and constructors not shown

    private String name;
    private boolean publicSchool;
    private int enrollment;
}

public class Mascot {

    boolean isAnimal() {
        return type == ANIMAL;
    }

    // other methods and constructors not
    // shown

    private String name;
    private int type;

    <*1> ANIMAL = 0;
    <*1> HUMAN = 1;
    <*1> COLOR = 2;
    <*1> OTHER = 3;
}
```

**QUESTION 10**

Which of the following replaces **<\*1>** in the code to the right so that the method checks whether `s1` is a subset of `s2`?

- A. `s2.add(iter.next())`
- B. `s2[iter.next()] == false`
- C. `s2[iter.next()]`
- D. `!s2.contains(iter.next())`
- E. None of these

```
public static boolean subset(Set s1,
                             Set s2) {
    Iterator iter = s1.iterator();
    while (iter.hasNext())
        if (<*1>)
            return false;
    return true;
}
```

**QUESTION 11**

Assume **<\*1>** has been filled in correctly. Which of the following checks whether `Set s1` and `Set s2` have the same elements?

- A. `subset(s1,s2) && subset(s2,s1)`
- B. `subset(s1,s2) || subset(s2,s1)`
- C. `s1 == s2`
- D. `s1.contains(s2).contains(s1)`
- E. More than one of these

**QUESTION 12**

What is returned by `mangle("television")`?

- A. television
- B. noisivelet
- C. tevnsisnse
- D. tensinfdel
- E. None of these

```
public static String mangle(String s) {
    StringBuffer sb = new StringBuffer(s);
    int len = sb.length();
    for (int i=0; i<len; ++i)
        sb.setCharAt(i,sb.charAt((i*i)%len));
    return sb.toString();
}
```

**QUESTION 13**

What is output by the code to the right?

- A. true -1                      B. false -1
- C. true 0                        D. false 0
- E. None of these

```
int x=0, y=0;

boolean b = (++x < y) & (y-- > 5);

System.out.print(b + " " + y);
```

**QUESTION 14**

Which of the following is not a subclass of `Object`?

- A. `String`      B. `TreeSet`      C. `Comparable`      D. `Integer`      E. None of these

**QUESTION 15**

Which of the following replaces **<\*1>** in the code to the right to make the `traverse()` method do a preorder traversal?

- A. `if (left != null) left.traverse();`  
`System.out.print(data);`  
`if (right != null) right.traverse();`
- B. `if (left != null) left.traverse();`  
`if (right != null) right.traverse();`  
`System.out.print(data);`
- C. `System.out.print(data);`  
`if (left != null) left.traverse();`  
`if (right != null) right.traverse();`
- D. `while (left) left.traverse();`  
`while (right) right.traverse();`  
`System.out.print(data);`
- E. None of these

For the remaining questions assume **<\*1>** has been filled in correctly.

**QUESTION 16**

Which of these creates a BST and adds an object representing the integer 3?

- A. `BST b;`  
`b.add(3);`
- B. `BST b;`  
`b.new();`  
`Integer i;`  
`i.new(3);`  
`b.add(i);`
- C. `BST b;`  
`Integer i;`  
`i=3;`  
`b.add(i);`
- D. `BST b;`  
`Integer i =`  
`new Integer(3);`  
`b.add(i);`
- E. None of these

**QUESTION 17**

Assume that  $n$  objects are added to a binary search tree in order from smallest to largest. What is the worst case running time for searching for an object in the tree? Choose the smallest correct answer.

- A.  $O(1)$
- B.  $O(\log n)$
- C.  $O((\log n)^2)$
- D.  $O(n)$
- E. None of these

```
public class BST {

    public BST(Comparable value) {
        data=value;
    }

    public BST add(Comparable value) {
        int c = data.compareTo(value);

        if (c<0) {
            if (right!=null)
                right.add(value);
            else right = new BST(value);
        }

        else if (c>0) {
            if (left!=null)
                left.add(value);
            else left = new BST(value);
        }

        return this;
    }

    public void traverse() {
        <*1>
    }
    // other methods not shown

    private BST left, right;
    private Comparable data;
}
```

**QUESTION 18**

Which of the following replaces <\*1> in the code to the right to insert an element into the circular list after the current element?

- A. 

```
CircleList c = new CircleList(o);
c.next = this.next;
c.previous = this;
this.next.previous = c;
this.next = c;
```
- B. 

```
CircleList c = new CircleList(o);
c.next = this;
c.previous = this.previous;
this.previous.next = c;
this.previous = c;
```
- C. 

```
CircleList c = new CircleList(o);
c.next = this;
c.previous = this;
this.next = c;
this.previous = c;
```
- D. 

```
CircleList c = new CircleList(o);
c.next = this.next;
c.previous = this.previous;
this.next.previous = c;
this.previous.next = c;
```
- E. None of these

**QUESTION 19**

Assume <\*1> has been filled in correctly. Which of the following replaces <\*2> in the code to the right to print each element of the circular list exactly once?

- A. 

```
while (cl != this) {
    System.out.print(cl.item);
    cl = cl.next; }
```
- B. 

```
do {
    System.out.print(cl.item);
    cl = cl.next;
} while (cl!=this);
```
- C. 

```
System.out.print(cl.item);
```
- D. 

```
for(int i=0; i<cl; ++i)
    System.out.print(cl[i].item);
```
- E. None of these

```
// Creates a circular doubly linked list
```

```
public class CircleList {

    public CircleList() {}
    public CircleList(Object o) {
        item = o;
        next = previous = this;
    }

    public CircleList insert(Object o) {
        if (next == null) {
            item = o;
            next = previous = this;
        }
        else {
            <*1>
        }
        return this;
    }

    public CircleList remove() {
        // code not shown
    }

    public void print() {
        if (next != null) {
            CircleList cl = this;
            <*2>
        }
    }

    private Object item;
    private CircleList next;
    private CircleList previous;

}
```

**QUESTION 20**

Which of the following returns true?

- A. `mystery("banana")`
- B. `mystery("abcdeedcba")`
- C. `mystery("ASDFDSA")`
- D. `mystery("ananabanana")`
- E. More than one of these

```
public static boolean mystery(String s) {
    int len = s.length();
    for (int i=0; i<len/2; ++i) {
        if (s.charAt(i) != s.charAt(len-i-1))
            return false;
        if (s.charAt(i) > s.charAt(i+1))
            return false;
    }
    return true;
}
```

**QUESTION 21**

`GregorianCalendar` is a subclass of `Calendar` that is not abstract. Which of these statements is true?

- A. The `GregorianCalendar` class does not have to override the `add()` method and can not override the `set()` method.
- B. The `GregorianCalendar` class can not override the `add()` method or the `set()` method.
- C. The `GregorianCalendar` class must override the `add()` method but not the `set()` method.
- D. The `GregorianCalendar` class must override the `add()` method and the `set()` method.
- E. None of these

```
// The following is part of the built-in
// class Calendar

package java.util;

public abstract class Calendar implements
    Cloneable, Serializable {

    protected Calendar() {
        // code not shown
    }

    protected Calendar(TimeZone zone,
        Locale aLocale) {
        // code not shown
    }

    public static Calendar getInstance() {
        // code not shown
    }

    public abstract void add(int field,
        int amount);

    public final void set(int field,
        int value) {
        // code not shown
    }

    // other methods and class constants
    // not shown
}
```

**QUESTION 22**

Assume the `java.util` package has been imported. Where would the following declaration be allowed?

```
Calendar c = new Calendar();
```

- A. Only in package `java.util`
- B. Only in subclasses of `Calendar`
- C. Package `java.util` and subclasses of `Calendar`
- D. Anywhere
- E. None of these

**QUESTION 23**

Which of the following is possible as the return value of `s1.compareTo(s2)` if `s1` and `s2` are lower case words with `s1` coming before `s2` in the dictionary?

- A. -5
- B. 0
- C. 1
- D. 4
- E. None of these

**QUESTION 24**

Which of the following replaces **<\*1>** in the code to the right to throw an appropriate exception?

- A. `throw Error();`
- B. `throw  
    ArrayIndexOutOfBoundsException;`
- C. `throw RuntimeException();`
- D. `throw IllegalArgumentException;`
- E. None of these

**QUESTION 25**

Which of the following replaces **<\*2>** in the code to the right to access the cost of the element in the list of components with index *i*?

- A. `((Component) comps.get(i)).cost()`
- B. `((Component) comps.get(i)).cost()`
- C. `(Component) (comps).get(i).cost()`
- D. `comps.get(i).cost()`
- E. More than one of these

**QUESTION 26**

Suppose the class `Monitor` is an implementation of the interface `Component`, and that `Monitor` has a no-argument constructor. Which of these declarations is valid?

- A. `Component c = new Monitor();`
- B. `Monitor m = new Component();`
- C. `Component c = new Component;`
- D. `Monitor m = new Monitor;`
- E. More than one of these

**QUESTION 27**

What function is computed by `f()`?

- A. Addition
- B. Subtraction
- C. Multiplication
- D. Division
- E. None of these

```
public interface Component {
    public double cost();
    public double weight();
}

public class Computer {

    public Computer() {
        comps = new ArrayList();
    }

    public Computer addComponent(Component c)
    {
        if (c == null) <*1>
        else {
            comps.add(c);
            return this;
        }
    }

    public double cost() {
        double sum = 0.0;
        for (int i=0; i<comps.size(); ++i)
            sum += <*2>;
        return 2.0*sum;
    }

    private ArrayList comps;
}
```

```
public static int f(int x, int y) {
    if (x == 0) return 0;
    else if (x < 0) return -f(-x, y);
    else return y + f(x-1, y);
}
```

**QUESTION 28**

How many times is "sc" compared to an element in A when executing `find(A, 0, 5, "sc")` where A is the array below?

"aa"	"ac"	"db"	"ee"	"sc"	"zw"
------	------	------	------	------	------

- A. 0                                      B. 2  
 C. 4                                      D. 6  
 E. None of these

```
// Implements binary search

public static boolean find(Comparable[] A,
                           int r, int s,
                           Comparable item) {

    if (r>s)
        return false;
    int mid = (r+s)/2;
    if (A[mid].equals(item)) return true;
    else if (A[mid].compareTo(item)<0)
        return find(A,mid+1,s,item);
    else
        return find(A,r,mid-1,item);
}
```

**QUESTION 29**

What is the maximum size of an array that can be searched with binary search using at most n comparisons with the item being searched for?

- A.  $2n$                                       B.  $n^2$   
 C.  $2^n$                                       D.  $2^n - 1$   
 E. None of these

**QUESTION 30**

Which of the following replaces <\*1> in the code to the right to make it perform as specified?

- A. `sum = 0;`  
 B. `return 0;`  
 C. `System.exit();`  
 D. No code is needed  
 E. None of these

```
// Returns the sum of two integers
// represented as strings. If either
// string is not a number, returns the
// other number. If both are not numbers
// returns 0

public static int add(String s1, String s2)
{
    int sum = 0;
    try {
        sum += Integer.parseInt(s1);
    }
    catch(NumberFormatException e) {
        <*1>
    }
    try {
        sum += Integer.parseInt(s2);
    }
    catch(NumberFormatException e) {
        <*1>
    }
    return sum;
}
```

**QUESTION 31**

Assume <\*1> has been filled in correctly. What is returned by `add("57", "-62")`?

- A. 0                                      B. -5  
 C. 57                                      D. -62  
 E. None of these



**QUESTION 32**

Which of these shows what array A looks like after the call process (A) , where A is the array below?

17	-3	24	-5	10	10
----	----	----	----	----	----

A.

17	3	24	5	10	10
----	---	----	---	----	----

B.

0	0	0	0	0	0
---	---	---	---	---	---

C.

17	-2	24	-4	10	10
----	----	----	----	----	----

D. A run time error occurs

E. None of these

```
public static void process(int[] A) {
    for (int i=0; i<A.length; ++i)
        if (A[i] < 0) A[i--]++;
}
```

**QUESTION 33**

What is output by output (7,3)?

- A. 0                                      B. 1  
C. 2                                      D. 3  
E. None of these

```
public static void output(int x, int y) {
    if (x == 0) return;
    else {
        switch(y%4) {
            case 0: System.out.print(0);
                    output(x/10, y*y);
            case 1: System.out.print(1);
                    output(x/10, y*y);
            case 2: System.out.print(2);
                    output(x/10, y*y);
            case 3: System.out.print(3);
                    output(x/10, y*y);
        }
    }
}
```

**QUESTION 34**

What is output by output (27, 2)?

- A. 20                                      B. 012301230123  
C. 2012330123                      D. 20000  
E. None of these

**QUESTION 35**

Which of these types would be a valid replacement for <\*1> in the code to the right so that the assignment shown can be done without a cast?

- A. Integer  
B. String  
C. byte  
D. long  
E. More than one of these

```
int x = 27;
<*1> y;
y = x;
```

**QUESTION 36**

A valid line of input contains a student's first name, last name, and id number all separated by one or more spaces. What replaces `<*1>`, `<*2>`, and `<*3>` in the code to the right to extract this input?

- A. All are replaced with `in.nextToken()`
- B. All are replaced with `st.nextToken()`
- C. `<*1>`: `in.token(1)`  
`<*2>`: `in.token(2)`  
`<*3>`: `in.token(3)`
- D. `<*1>`: `st.getToken(1)`  
`<*2>`: `st.getToken(2)`  
`<*3>`: `st.getToken(3)`
- E. None of these

**QUESTION 37**

The `Student` class does not override the `equals()` method inherited from `Object`. When are two instances of `Student` equal according to the `equals()` method?

- A. When they have the same reference
- B. When their `firstname` fields have the same reference, their `lastname` fields have the same reference and their `id` fields are the same
- C. When their `firstname` fields are equal using the `equals()` overridden by `String`, their `lastname` fields are equal using the `equals()` overridden by `String`, and their `id` fields are the same
- D. Never
- E. Both A and B

**QUESTION 38**

What does matrix A look like after the call `rearrange(A)` where A is the matrix below?

1	2	3
4	5	6
7	8	9

- A. 

1	4	7
2	5	8
3	6	9
- B. 

7	8	9
4	5	6
1	2	3
- C. 

1	2	3
4	5	6
7	8	9
- D. 

3	2	1
6	5	4
9	8	7
- E. None of these

```
public class Student {

    // Assume IO.readLine() reads one line
    // of input from the keyboard and returns
    // it as a String

    public void input() {
        String in = IO.readLine();
        StringTokenizer st =
            new StringTokenizer(in);
        firstname = <*1>;
        lastname = <*2>;
        id = Integer.parseInt(<*3>);
    }

    // other methods not shown

    private String firstname, lastname;
    private int id;
}
```

```
public static void rearrange(int[][] A) {
    int[] temp;
    int[][] B = new int[A.length][];
    for (int i=0; i<A.length; ++i) {
        B[i] = A[A.length - i - 1];
    }
    A = B;
}
```

**QUESTION 39**

What replaces **<\*1>** in the code to the right to select the substring of `big` starting at index `i` with length `smallllen`?

- A. `big.substring(i, smallllen)`
- B. `big.substring(i, i+smallllen)`
- C. `big.substring(i-1, smallllen)`
- D. `big.substring(i, i+smallllen-1)`
- E. None of these

**QUESTION 40**

Assume **<\*1>** has been filled in correctly. What is returned by `count("abracadabra", "abr")`?

- A. 2
- B. 3
- C. 4
- D. 5
- E. None of these

```
public static int count(String big,
                        String small) {
    int biglen = big.length();
    int smallllen = small.length();

    int total = 0;

    for (int i=0; i<biglen-smallllen; ++i)
        if (small.equals(<*1>)) ++total;

    return total;
}
```

# Computer Science Answer Key

## UIL State 2004

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