Practice Test 1

AP® Computer Science A Exam

SECTION I: Multiple-Choice Questions

DO NOT OPEN THIS BOOKLET UNTIL YOU ARE TOLD TO DO SO.

At a Glance

Total Time
1 hour 30 minutes
Number of Questions
40
Percent of Total Score
50%
Writing Instrument
Pencil required

Instructions

Section I of this examination contains 40 multiple-choice questions. Fill in only the ovals for numbers 1 through 40 on your answer sheet.

Indicate all of your answers to the multiple-choice questions on the answer sheet. No credit will be given for anything written in this exam booklet, but you may use the booklet for notes or scratch work. After you have decided which of the suggested answers is best, completely fill in the corresponding oval on the answer sheet. Give only one answer to each question. If you change an answer, be sure that the previous mark is erased completely. Here is a sample question and answer.

Sample Question   Sample Answer

Chicago is a   Ⓐ ● Ⓒ Ⓓ Ⓔ
(A) state
(B) city
(C) country
(D) continent
(E) county

Use your time effectively, working as quickly as you can without losing accuracy. Do not spend too much time on any one question. Go on to other questions and come back to the ones you have not answered if you have time. It is not expected that everyone will know the answers to all the multiple-choice questions.

About Guessing

Many candidates wonder whether or not to guess the answers to questions about which they are not certain. Multiple-choice scores are based on the number of questions answered correctly. Points are not deducted for incorrect answers, and no points are awarded for unanswered questions. Because points are not deducted for incorrect answers, you are encouraged to answer all multiple-choice questions. On any questions you do not know the answer to, you should eliminate as many choices as you can, and then select the best answer among the remaining choices.

GO ON TO THE NEXT PAGE.

Quick Reference

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

class java.lang.Integer
- Integer(int value)
- int intValue()
- Integer.MIN_VALUE   // minimum value represented by an int or Integer
- Integer.MAX_VALUE   // maximum value represented by an int or Integer

class java.lang.Double
- Double(double value)
- double doubleValue()

class java.lang.String
- int length()
- String substring(int from, int to)   // returns the substring beginning at from
                                             // and ending at to-1
- String substring(int from)   // returns substring(from, length())
- int indexOf(String str)   // returns the index of the first occurrence of str;
                                             // returns -1 if not found
- int compareTo(String other)   // returns a value < 0 if this is less than other
                                             // returns a value = 0 if this is equal to other
                                             // returns a value > 0 if this is greater than other

class java.lang.Math
- static int abs(int x)
- static double abs(double x)
- static double pow(double base, double exponent)
- static double sqrt(double x)
- static double random()   // returns a double in the range [0.0, 1.0)

interface java.util.List<E>
- int size()
- boolean add(E obj)   // appends obj to end of list; returns true
- void add(int index, E obj)   // inserts obj at position index (0 ≤ index ≤ size),
                                             // moving elements at position index and higher
                                             // to the right (adds 1 to their indices) and adjusts size
- E get(int index)
- E set(int index, E obj)   // replaces the element at position index with obj
                                             // returns the element formerly at the specified position
- E remove(int index)   // removes element from position index, moving elements
                                             // at position index + 1 and higher to the left
                                             // (subtracts 1 from their indices) and adjusts size
                                             // returns the element formerly at the specified position

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

GO ON TO THE NEXT PAGE.

COMPUTER SCIENCE A

SECTION I

Time—1 hour and 30 minutes

Number of Questions—40

Percent of total exam grade—50%

Directions: Determine the answer to each of the following questions or incomplete statements, using the available space for any necessary scratchwork. Then decide which is the best of the choices given and fill in the corresponding oval on the answer sheet. No credit will be given for anything written in the examination booklet. Do not spend too much time on any one problem.

Notes:
- Assume that the classes listed in the Quick Reference have been imported where appropriate.
- Assume that declarations of variables and methods appear within the context of an enclosing class.
- Assume that method calls that are not prefixed with an object or class name and are not shown within a complete class definition appear within the context of an enclosing class.
- Unless otherwise noted in the question, assume that parameters in the method calls are not `null` and that methods are called only when their preconditions are satisfied.

MULTIPLE CHOICE QUESTIONS   **USE THIS SPACE FOR SCRATCHWORK**

1. Consider the following methods.

```
public void trial()
{
    int a = 10;
        int b = 5;
        doublevalues(a,b);
        System.out.print(b);
    System.out.print(a);
}

public void doublevalues(int c, int d)
{
        c = c * 2;
        d = d * 2;
    System.out.print(c);
    System.out.print(d);
}
```

What is printed as the result of the call `trial()`?
(A) `2010`
(B) `2010105`
(C) `2010510`
(D) `20102010`
(E) `20101020`

GO ON TO THE NEXT PAGE.

2. Consider the following method.

```
/**
 * Precondition: a > b > 0
 */
public static int mystery(int a, int b)
{
    int d = 0;
    for (int c = a; c > b; c--)
    {
        d=d+c;
    }
    return d;
}
```

What is returned by the call `mystery(x,y)`?

(A) The sum of all the integers greater than y but less than or equal to x
(B) The sum of all the integers greater than or equal to y but less than or equal to x
(C) The sum of all the integers greater than y but less than x
(D) The sum of all the integers greater than y but less than or equal to x
(E) The sum of all the integers less than y but greater than or equal to x

3. Consider the following method.

```
public void mystery (int  n)
{
    int k;
    for (k = 0 ; k < n ; k++)
    {
        mystery(k);
        System.out.print (n) ;
    }
}
```

What value is returned by the call `mystery(3)` ?

(A) `0123`
(B) `00123`
(C) `0010012`
(D) `00100123`
(E) `001001200100123`

4. Consider an array of integers.

| 4 | 10 | 1 | 2 | 6 | 7 | 3 | 5 |

If selection sort is used to order the array from smallest to largest values, which of the following represents a possible state of the array at some point during the selection sort process?

(A)	1	4	10	2	3	6	7	5
(B)	1	2	4	6	10	7	3	5
(C)	1	2	3	10	6	7	4	5
(D)	4	3	1	2	6	7	10	5
(E)	5	3	7	6	2	1	10	4

GO ON TO THE NEXT PAGE.

5. Consider the following code segment:

```
int   k;
int   A[];
A = new int [7];
for (k = 0; k < A.length; k++)
{
    A[k] = A.length - k;
}
for (k = 0; k < A.length - 1; k++)
{
    A[k+1] = A[k];
}
```

What values will A contain after the code segment is executed?

(A) 1	1	2	3	4	5	6
(B) 1	2	3	4	5	6	7
(C) 6	6	5	4	3	2	1
(D) 7	7	6	5	4	3	2
(E) 7	7	7	7	7	7	7

```
int   k;
int   A[];
A = new int [7];
```

Questions 6–7 refer to the following two classes.

```
public class PostOffice
{
    // constructor initializes boxes
    // to length 100
    public PostOffice( )
    {   /* implementation not shown   */}

    // returns the given p.o. box
    //  0 <= theBox  <  getNumBoxes ()
    public Box getBox (int theBox)
    {   /* implementation not shown   */}
    // returns the number of p.o. boxes
    public int getNumBoxes ()
    {   /* implementation not shown   */}

    // private data members and
    // other methods not shown
}


public class Box
{
    // constructor
    public Box ( )
    {   /* implementation not shown  */}

    // returns the number of this box
    public int getBoxNumber ( )
    {   /* implementation not shown   */}

    // returns the number of pieces
    // of mail in this box
    public int getMailCount ( )
    {   /* implementation not shown  */}
    // returns the given piece of mail
    // 0 <= thePiece < getMailCount ( )
    public Mail getMail (int thePiece)
    {   /* implementation not shown  */}
    // true if the box has been assigned
    // to a customer
    public boolean isAssigned ( )
    {   /* implementation not shown  */}
    // true if the box contains mail
    public boolean hasMail ( )
    {   /* implementation not shown  */}
    // private data members and
    // other methods not shown
}
public class Mail
{
    // private members,  constructors, and
    // other methods not shown
}
```

GO ON TO THE NEXT PAGE.

6. Consider the following code segment:

```
PostOffice p[ ]
p = new PostOffice[10];
```

Assuming that the box has been assigned and that it has at least four pieces of mail waiting in it, what is the correct way of getting the fourth piece of mail from the 57th box of the tenth post office of p?

(A) `Mail m = p[10].getBox(57).getmail(4);`

(B) `Mail m = p[9].getBox(56).getMail (3);`

(C) `Mail m = p.getMail(57).getMail (4) [10];`

(D) `Mail m = getMail(getBox(p[9], 560, 3);`

(E) `Mail m = new Mail(10, 57, 4);`

GO ON TO THE NEXT PAGE.

7. Consider the incomplete function `printEmptyBoxes` given below. `printEmptyBoxes` should print the box numbers of all of the boxes that do not contain mail.

```
public void printEmptyBoxes (PostOffice p [ ])
{
   for (int k = 0; k < p.length - 1 ; k++)
   {
      for (int x = 0; x < p[k].getNumBoxes( ) - 1 ; x++)
      {
           //  missing expression
      }
   }
}
```

Which of the following could be used to replace

```
// missing expression
```

body so that `printBoxesWithoutMail` works as intended?

(A)
```
if (p[k].getBox(x).isAssigned( ) &&
    !p[k].getBox(x).hasMail( ) )
    {
        System.out.println(P[k].getBox(x).getBoxNumber( )) ;
    }
```
(B)
```
if (p[x].getBox(k).isAssigned ( ) &&
    !p[x].getBox(k).hasMail( ))
    {
      System.out.println(p[x].getBox(k).getBoxNumber( ));
    }
```
(C)
```
if (p[k].getBox(x).isAssigned( ) &&
    !p[k].getBox(x).hasMail( ))
    {
        System.out.println (p[k].getBoxNumber (x));
    }
```
(D)
```
if (p[x].getBox(k).isAssigned( ) &&
    !p[x].getBox (k).hasMail( ))
    {
        System.out.println(p[x].getBoxNumber(k));
     }
```
(E)
```
if (p[x].getBox(k).isAssigned( ) &&
    p[x].getBox(k).getMail( )   ==   0)
    {
        System.out.println(k);
    }
```

GO ON TO THE NEXT PAGE.

8. Assume that a and b are Boolean variables that have been initialized. Consider the following code segment.

```
a = a && b;
b = a || b;
```

Which of the following statements is true?

   I. The final value of a is the same as the initial value of a.
  II. The final value of b is the same as the initial value of b.
 III. The final value of a is the same as the initial value of b.

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

9. Consider the following code segment.

```
int x;
x = 53;
if (x > 10)
{
    System.out.print("A") ;
}
if (x > 30)
{
    System.out.print("B") ;
}
else if (x > 40)
{
    System.out.print("C") ;
}
if (x > 50)
{
    System.out.print ("D") ;
}
if (x > 70)
{
    System.out.print ("E") ;
}
```

What is output when the code is executed?

(A) A
(B) D
(C) ABD
(D) ABCD
(E) ABCDE

GO ON TO THE NEXT PAGE.

10. Consider the following code segment:

```
int j;
int k;
for (j = -2; j <= 2; j = j + 2)
{
    for (k = j; k < j + 3; k++)
    {
        System.out.print(k + " " ) ;
    }
}
```

What is the output when the code is executed?

(A) -2   -1   0
(B) -2   -1   0   1   2
(C) 0   1   2   0   1   2   0   1   2
(D) -2   0   2
(E) -2   -1   0   0   1   2   2   3   4

11. Consider the following method.

```
public void mystery (int  count, String s)
{
    if  (count  <=  0)
    {
        return;
    }
    if  (count % 3 == 0)
    {
        System.out.print(s + "--" + s)
    }
    else if (count %  3 == 1)
    {
        System.out.print(s + "-" + s)
    }
    else
    {
        System.out.print (s) ;
    }
    mystery(count - 1, s) ;
}
```

What is output by the call `mystery(5, "X");`?

(A) XX - XX - - XXX - X
(B) XX - XX - XX - XX
(C) XXX - - XX - X - XX - - XXX
(D) XX - XXX - - XXX - XX
(E) XXXXX

Questions 12–13 refer to the following classes and method descriptions.

Class `Table` has a method, `getPrice`, which takes no parameters and returns the price of the table.

Class `Chair` also has a method, `getPrice`, which takes no parameters and returns the price of the chair.

Class `DiningRoomSet` has a constructor which is passed a `Table` object and an `ArrayList` of `Chair` objects. It stores these parameters in its private data fields `myTable` and `myChairs`.

Class `DiningRoomSet` has a method, `getPrice`, which takes no parameters and returns the price of the dining room set. The price of a dining room set is calculated as the sum of the price of its table and all of its chairs.

12. What is the correct way to define the signature of the constructor for the `DiningRoomSet` class?

(A) `public void DiningRoomSet (Table t, ArrayList, chairs)`
(B) `public DiningRoomSet (Table t, ArrayList chairs)`
(C) `public void DiningRoomSet (Table t, ArrayList Chair Chairs)`
(D) `public DiningRoomSet (Table t, ArrayList Chair Chairs)`
(E) `public DiningRoomSet (Table t, Chair Chairs)`

13. What is the correct way to implement the getPrice method of the `DiningRoomSet` class?

(A)
```
public double getPrice(Table t, ArrayList chairs)
{
    return t.getPrice() + chairs.getPrice() ;
}
```
(B)
```
public double getPrice(Table t, ArrayList chairs)
{
    return myTable.getPrice() + myChairs.getPrice();
}
```
(C)
```
public double getPrice()
{
    return myTable.getPrice() + myChairs.getPrice();
}
```
(D)
```
public double getPrice()
{
    double result  = myTable.getPrice();
    for (int k = 0; k < myChairs.size() - 1; k++)
    {
        result += ((Chair)myChairs.get(k)).getPrice();
    }
    return result;
}
```
(E)
```
public double getPrice()
{
    double result = myTable.getPrice() ;
    for (int k = 0; k  <  myChairs.length - 1; k++)
    {
        result += ((Chair)myChairs[k]).getPrice( );
    }
    return result ;
}
```

GO ON TO THE NEXT PAGE.

14. Consider the following output:

```
6    5    4    3    2    1
5    4    3    2    1
4    3    2    1
3    2    1
2    1
1
```

Which of the following code segments produces the above output when executed?

(A)
```
for  (int j = 6;  j  <  0;  j--)
    {
      for (int k = j; k > 0 ; k --)
      {
          System.out.print (k  +  "    ") ;
      }
      System.out.print1n("") ;

    }
```
(B)
```
for (int j = 6; j >= 0; j--)
    {
      for (int k = j; k >= 0; k--)
      {
          System.out.print(k + " ");
      }
      System.out.print1n (" ") ;

    }
```
(C)
```
for (int j = 0; j < 6; j++)
    {
      for (int k = 6 - j; k > 0; k--)
      {
          System.out.print (k + "  ") ;
      }
      system.out.print1n (" ") ;

    }
```
(D)
```
for (int j = 0; j < 6; j++)
    {
      for (int k = 7 - j ; k > 0 ; k --)
      {
          System.out.print(k + " ");
      }
      System.out.print1n (" ") ;

    }
```
(E)
```
for (int j = 0; j < 6; j++)
    {
      for (int k = 6  -  j ; k >= 0; k--)
      {
          System.out.print(k + " ");
      }
      System.out.print1n (" ") ;

    }
```

15. Consider the following code segment.

```
List<Integer> list = new ArrayList<Integer>();
list.add(new Integer (7));
list.add (new Integer (6));
list.add (1, new Integer (5));
list.add (1, new Integer (4));
list.add (new Integer (3));
list.set (2, new Integer (2));
list.add (1, new Integer (1));
System.out.prinln (list);
```

What is printed as a result of executing this code segment?

(A) [1, 4, 2, 7, 6, 3]
(B) [7, 1, 4, 2, 6, 3]
(C) [7, 2, 5, 4, 3, 1]
(D) [7, 6, 2, 4, 3, 1]
(E) [7, 1, 2]

16. Consider the following declarations.

```
public interface Animal
{
    String makeSound();
    String animalType();
}

    public static class Dog implements Animal
{
    public String makeSound(Animal a)
    {
       // Implementation not shown
    }
}
```

Which of the following methods must be included in the declaration of the Dog class in order for the class to successfully compile?

I.   public String makeSound()
II. public String animalType()
III. public String animalType(animal b)

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

GO ON TO THE NEXT PAGE.

17. Consider the following two classes.

```
public static class Fish
{
  public String endoskeleton = "bone";

  public void action()
  {
      System.out.println("splash splash");
  }
}

public static class Shark extends Fish
{
  public void action()
  {
      System.out.println("chomp chomp");
  }

  public String endoskeleton="cartilage";
}
```

Which of the following is the correct output after the following code segment is executed?

```
Fish Bob = new Shark();
System.out.println(Bob.endoskeleton);
Bob.action();
```

(A) bone
    chomp chomp
(B) bone
    splash splash
(C) cartilage
    splash splash
(D) cartilage
    chomp chomp
(E) cartilage
    splash splash
    chomp chomp

GO ON TO THE NEXT PAGE.

Questions 18–19 refer to the following incomplete method.

18. The following insertSort method sorts the values in an integer array, sort, in ascending order.

```
Line 1:    public static void insertSort(int[] sort)
Line 2:        {
Line 3:            for (int index=1;index<sort.length;index++)
Line 4:                {
Line 5:        int temp=sort[index];
Line 6:        while (index > 0 && sort[index-1]>temp)
Line 7:                    {
Line 8:            // Missing code
Line 9:                }
Line 10:        sort[index]=temp;
Line 11:            }
Line 12:        }
```

Which of the following can be used to replace "// Missing code" so that the insertSort method will execute properly?

(A) `sort[index]=sort[index-1];`
    `index++;`
(B) `sort[index-1]=sort[index];`
    `index--;`
(C) `sort[index]=sort[index+1];`
    `index++;`
(D) `sort[index]=sort[index-1];`
    `index--;`
(E) `sort[index]=sort[index+1];`
    `index--;`

19. Assuming that the "// Missing code" is implemented properly, what change can be made to the code in order for the array to be sorted in descending order?

(A) Replace Line 6 with: `while (index < 0 && sort[index-1]<temp)`
(B) Replace Line 6 with: `while (index < 0 && sort[index-1]<temp)`
(C) Replace Line 6 with: `while (index > 0 && sort[index-1]<temp)`
(D) Replace Line 3 with: `for (int index=sort.length-1;index>0;index--)`
(E) Replace Line 3 with: `for (int index=1;index>0;index--)`

20. Which of the following arrays would be sorted the slowest using insertion sort?

(A) `[3 4 6 2 7 3 9]`
(B) `[3 2 5 4 6 7 9]`
(C) `[9 7 6 5 4 3 2]`
(D) `[2 3 4 5 6 7 9]`
(E) `[9 3 2 4 5 7 6]`

GO ON TO THE NEXT PAGE.

Questions 21–23 refer to the following incomplete class declaration used to represent fractions with integral numerators and denominators.

```
public class Fraction
{
 private int numerator ;
 private int denominator ;

 public Fraction ( )
 {
    numerator = 0 ;
    denominator = 1 ;
 }

 public Fraction (int n, int d)
 {
    numerator = n;
    denominator = d;
 }

 // postcondition: returns the
 //  numerator
 public int getNumerator ( )
 {  /* implementation not shown */  }

 // postcondition: returns the
 //  denominator
 public int getDenominator ( )
 {  /* implementation not shown*/   }

 // postcondition: returns the greatest
 // common divisor of x and y
 public int gcd (int x, int y)
 {  /* implementation not shown*/   }

 // postcondition: returns the Fraction
 //  that is the result of multiplying
 //  this Fraction and f

 public Fraction multiply (Fraction f)
 {  /* implementation not shown */  }
 // . . . other methods not shown
}
```

GO ON TO THE NEXT PAGE.

21. Consider the method multiply of the Fraction class.

```
// postcondition: returns the Fraction
//     that is the result of multiplying
//     this Fraction and f
public Fraction multiply (Fraction f)
{   /* missing code*/     }
```

Which of the following statements can be used to replace /* missing code */ so that the muliply method is corretly implemented?

```
I.    return Fraction (
        numerator * f.getNumerator ( ) ,
        denominator * f.getDenominator ( ) );
II.   return new Fraction (
        numerator * f.numerator ,
        denominator * f.denominator ( ) );
III.  return new Fraction (
        numerator * f.getNumerator ( ) ,
        denominator * f.getDenominator ( ) );
```

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

22. Consider the use of the Fraction class to muliply the fractions $\frac{3}{4}$ and $\frac{7}{19}$. Consider the following code:

```
Fraction fractionOne;
Fraction fractionTwo;
Fraction answer;
fractionOne = new Fraction (3, 4) ;
fractionTwo = new Fraction (7, 19) ;
/* missing code */
```

Which of the following could be used to replace /* missing code */ so that answer contains the result of muliplying fractionOne by fractionTwo?

```
(A) answer = fractionOne * fractionTwo
(B) answer = multiply (fractionOne,fractionTwo) ;
(C) answer = fractionOne.multiply (fractionTwo) ;
(D) answer = new Fraction (fractionOne,fractionTwo) ;
(E) answer = (fractionOne .getNumerator ( ) * fractionTwo .getNumerator ( ) ) /
        (fractionOne .getDenominator ( ) * fractionTwo .getDenominator ( ) ) ;
```

GO ON TO THE NEXT PAGE.

23. The following incomplete class declaration is intended to extend the `Fraction` class so that fractions can be manipulated in reduced form (lowest terms).

Note that a fraction can be reduced to lowest terms by dividing both the numerator and denominator by the greatest common divisor (gcd) of the numerator and denominator.

```
public class ReducedFraction extends Fraction
{
    private int reducedNumerator ;
    private int reducedDenominator ;
    // . . . constructors and other methods not shown
}
```

Consider the following proposed constructors for the `ReducedFraction` class:

```
I.   public ReducedFraction ( )
     {
       reducedNumerator = 0;
       reducedDenominator = 1;
     }
II.  public reducedFraction (int n, int d)
     {
       numerator = n;
       denominator = d;
       reducedNumerator = n / gcd (n, d);
       reducedDenominator = d / gcd (n, d);
     }
III. public ReducedFraction (int n, int d)
     {
       super (n, d) ;
       reducedNumerator = n / gcd (n, d) ;
       reducedDenominator = d / gcd (n, d) ;
     }
```

Which of these constructor(s) would be legal for the `ReducedFraction` class?

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

GO ON TO THE NEXT PAGE.

24. Consider s1 and s2 defined as follows.

```
String s1 = new String("hello") ;
String s2 = new String("hello") ;
```

Which of the following is/are correct ways to see if s1 and s2 hold identical strings?

```
I.   if (s1 == s2)
        /* s1 and s2 are identical */
II.  if (s1 .compareTo (s2) == 0 )
        /* s1 and s2 are identical */
III. if (s1 .equals (s2) )
        /* s1 and s2 are identical */
```

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

25. Consider the following variable and method declarations:

```
String s ;
String t ;
public void mystery (String a, String b)
{
    a = a + b ;
    b = b + a ;
}
```

Assume that s has the value "Elizabeth" and t has the value "Andrew" and mystery (s, t) is called. What are the values of s and t after the call to mystery?

	a	b
(A)	Elizabeth	Andrew
(B)	ElizabethAndrew	AndrewElizabeth
(C)	ElizabethAndrew	AndrewElizabethAndrew
(D)	ElizabethAndrew	ElizabethAndrewAndrew
(E)	ElizabethAndrewElizabeth	AndrewElizabethAndrew

GO ON TO THE NEXT PAGE.

26. Consider the following incomplete and *incorrect* class declaration:

```
public class Point implements Comparable
{
    private int x;
    private int y;
    public boolean compareTo (Point other)
    {
        return (x == other.x &&
                y == other.y);
    }
    // . . . constructors and other methods
    // not shown
}
```

For which of the following reasons is the above class declaration incorrect?

I.   Objects may not access private data fields of other objects in the same class.
II. The Comparable interface requires that compareTo be passed an Object rather than a Point.
III. The Comparable interface requires that compareTo return an int rather than a boolean.

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

GO ON TO THE NEXT PAGE.

27. Consider the following abstraction of a `for` loop where `<1>`, `<2>`, `<3>`, and `<4>` represent legal code in the indicated locations:

```
for (<1>; <2>; <3>)
{
    <4>
}
```

Which of the following `while` loops has the same functionality as the above `for` loop?

(A)
```
<1> ;
while (<2>)
{
    <3>;
    <4>
}
```
(B)
```
<1> ;
while ( <2>)
{
    <4>
    <3> ;
}
```
(C)
```
<1> ;
    while ( !<2>)
{
    <3> ;
    <4>
}
```
(D)
```
<1> ;
while ( !<2>)
{
    <4>
    <3>;
}
```
(E)
```
<1> ;
<3> ;
while ( <2>)
{
    <4>
    <3> ;
}
```

28. Consider the following expression:

```
a / b + c - d % e * f
```

Which of the expressions given below is equivalent to the one given above?

(A) `( (a / b) + (c - d) ) % (e *f)`
(B) `( ( ( ( a / b) + c) - d) % e) * f`
(C) `( ( a / b) + c) - (d % (e * f)`
(D) `(a / ( (b + c) - d)  % e) * f`
(E) `( (a / b) + c) - ( (d % e) * f)`

29. Assume that a program declares and initializes x as follows:

```
String [] x ;
x = new String[10] ;
initialize(x);              // Fills the array x with
                            // valid strings each of
                            // length 5
```

Which of the following code segments correctly traverses the array and prints out the first character of all ten strings followed by the second character of all ten strings, and so on?

```
I. int i;
   int j;
   for (i = 0 ; i < 10 ; i++)
        for (j = 0 ; j < 5 ; j++)
            System.out.print (x[i].substring (j, j+1));
II.  int i;
     int j;
     for (i = 0 ; i < 5 ; i++)
        for (j = 0 ; j < 10 ; j++)
            System.out.print (x[j].substring (i, i+1));
III. int i ;
     int j ;
     for (i = 0 ; i < 5 ; i++)
        for (j = 0 ; j < 10 ; j++)
            System.out.print (x[i].substring (j, j+1));
```

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

30. Consider the following declaration and assignment statements:

```
int a = 7 ;
int b = 4 ;
double c ;
c = a / b ;
```

After the assignment statement is executed, what's the value of c?

(A) 1.0
(B) 1.75
(C) 2.0
(D) An error occurs because c was not initialized.
(E) An error occurs because because a and b are integers and c is a double.

31. Consider the following code segment:

```
int x ;
x = /* initialized to an integer */
if (x % 2 == 0 && x / 3 == 1)
    System.out.print("Yes") ;
```

For what values of x will the word "Yes" be printed when the code segment is executed?

(A) 0

(B) 4

(C) Whenever x is even and x is not divisible by 3

(D) Whenever x is odd and x is divisible by 3

(E) Whenever x is even and x is divisible by 3

32. Consider the following incomplete class definition:

```
public class SomeClass
{
    private String myName ;
    // postcondition: returns myName
    public String getName ( )
    { /* implmentation not shown */ }
    // postcondition: myName == name
    public void setName (String name)
    { /* implmentation not shown */ }
    // . . . constructors, other methods
    // and private data not shown
}
```

Now consider the method swap, not part of the SomeClass class.

```
// precondition: x and y are correctly
// constructed
// postcondition: the names of objects
// x and y are swapped
public void swap (SomeClass x, SomeClass y)
{
    <missing code>
}
```

GO ON TO THE NEXT PAGE.

Which of the following code segments can replace <missing code> so that the method swap works as intended?

```
I.    SomeClass temp ;
      temp = x;
      x = y ;
      y = temp ;
II.   String temp ;
      temp = x.myName ;
      x .myName = y .myName
      y .myName = temp ;
III.  String temp ;
      temp = x .getName ( ) ;
      x .setName (y .getName ( ) ) ;
      y .setName (temp) ;
```

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

33. A bookstore wants to store information about the different types of books it sells.

For each book, it wants to keep track of the title of the book, the author of the book, and whether the book is a work of fiction or nonfiction.

If the book is a work of fiction, then the bookstore wants to keep track of whether it is a romance novel, a mystery novel, or science fiction.

If the book is a work of nonfiction, then the bookstore wants to keep track of whether it is a biography, a cookbook, or a self-help book.

Which of the following is the best design?

(A) Use one class, `Book`, which has three data fields: `String title`, `String author`, and `int bookType`.
(B) Use four unrelated classes: Book, Title, Author, and BookType.
(C) Use a class `Book` which has two data fields: `String title`, `String author`, and a subclass: `BookType`.
(D) Use a class `Book` which has two data fields: `String title`, `String author`, and two subclasses: `RomanceNovel, Mystery, ScienceFiction, Biography, Cookbook,` and `SelfHelpBook`.
(E) Use a class `Book` which has two data fields: `String title`, `String author`, and two subclasses: `FictionWork` and `NonFictionWork`. The class `FictionWork` has three subclasses, `RomanceNovel, Mystery,` and `ScienceFiction`. The class `NonFictionWork` has three subclasses: `Biography, Cookbook,` and `SelfHelpBook`.

GO ON TO THE NEXT PAGE.

34. Consider the following code:

```
public int mystery (int x)
{
    if (x == 1)
        return <missing value> ;
    else
        return (2 * mystery (x-1) ) + x ;
}
```

Which of the following can be used to replace <missing value> so that mystery (4) returns 34?

(A) 0
(B) 1
(C) 2
(D) 3
(E) 4

35. Consider the following code segment:

```
int [ ] X;
int [ ] Y;
X = initializeX ( ) ;    // returns a valid
                         // initialized int [ ]
Y = initializeY ( ) ;    // returns a valid
                         // initialized int [ ]
for (int k = 0 ;
   k < X.length && X[k] = = Y[k];
   k++)
{
    /* some code */
}
```

Assuming that after X and Y are initialized, X.length == Y.length, which of the following must be true after executing this code segment?

(A) k < X.length
(B) k < X.length && X[k] == Y[k]
(C) k < X.length && X[k] ! = Y[k]
(D) k >= X.length | | X[k] == Y[k]
(E) k >= X.length | | X[k] ! = Y[k]

36. Which of the following would *not* cause a run-time exception?

(A) Dividing an integer by zero
(B) Using an object that has been declared but not instantiated
(C) Accessing an array element with an array index that is equal to the length of the array
(D) Attempting to cast an object to a subclass of which it is not an instance
(E) Attempting to call a method with the wrong number of arguments

GO ON TO THE NEXT PAGE.

37. Assume that a and b are properly initialized variables of type `Double`.

    Which of the following is an equivalent expression to:

    ```
    a.doubleValue ( ) ! = b.doubleValue ( )
    ```

    (A) `a ! = b`
    (B) `a.notEquals (b)`
    (C) `! (a.doubleValue ( ) .equals(b.doubleValue ( ) ) )`
    (D) `! (a.compareTo(b) )`
    (E) `a.compareTo(b) ! = 0`

38. Which of the following would be the least effective way of ensuring reliability in a program?

    (A) Encapsulating functionality in a class by declaring all data fields to be public
    (B) Defining and following preconditions and postconditions for every method
    (C) Including assertions at key places in the code
    (D) Using descriptive variable names
    (E) Indenting code in a consistent and logical manner

39. Consider a dictionary that has 1,000 pages with 50 words on each page.

    In order to look up a given target word, a student is considering using one of the following three methods:

    Method 1

    Use a binary search technique to find the correct page (comparing the target word with the first word on a given page). When the correct page is found, use a sequential search technique to find the target word on the page.

    Method 2

    Use a sequential search technique to find the correct page (comparing the target word with the first word on a given page). When the correct page is found, use another sequential search technique to find the target word on the page.

    Method 3

    Use a sequential search technique on all of the words in the dictionary to find the target word.

    Which of the following best characterizes the greatest number of words that will be examined using each method?

	Method 1	Method 2	Method 3
(A)	10	50	1,000
(B)	55	500	2,500
(C)	55	525	25,000
(D)	60	1,050	1,050
(E)	60	1,050	50,000

GO ON TO THE NEXT PAGE.

40. Which of the following is *not* a peripheral?

    (A) A color laser printer
    (B) A monitor
    (C) A word processing application
    (D) A mouse
    (E) An external CD-ROM drive

END OF SECTION I

**IF YOU FINISH BEFORE TIME IS CALLED,
YOU MAY CHECK YOUR WORK ON THIS SECTION.**

DO NOT GO ON TO SECTION II UNTIL YOU ARE TOLD TO DO SO.

COMPUTER SCIENCE A

SECTION II

Time—1 hour and 30 minutes

Number of Questions—4

Percent of Total Grade—50%

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA™.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.

- Unless otherwise noted in the question, assume that parameters in method calls are not null and that methods are called only when their preconditions are satisfied.

- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

FREE RESPONSE QUESTIONS

1. In a certain school, students are permitted to enroll in one elective class from a list of electives offered. Because there are a limited number of spaces in each class for students, and because some electives are more popular than others, a lottery system was devised by the school to assign students to electives.

   Each student lists three choices for electives. The school orders the students randomly and assigns each student to the first available elective in the student's list of three choices. If none of the three choices is available (because those electives are fully enrolled), the school does not assign the student to an elective.

   After the school attempts to assign all of the students to electives, it produces a list of students it was unable to assign.

   For example, suppose there are six electives available to students: Astronomy, Ballroom Dance, Basketweaving, Constitutional Law, Marine Biology, and Programming.

   The following table shows the name, maximum enrollment, and current enrollment for six electives after 64 students have been successfully assigned to electives:

Elective Name	Maximum Enrollment	Current Enrollment
Astronomy	12	12
Ballroom Dance	20	3
Basketweaving	15	14
Constitutional Law	10	7
Marine Biology	10	10
Programming	30	30

GO ON TO THE NEXT PAGE.

Note that three elctives, Astronomy, Programming, and Marine Biology, are fully enrolled and are no longer options for students.

Now suppose that the following students need to be assigned to electives:

Student	First Choice getChoice (0)	Second Choice getChoice (1)	Third Choice getChoice (2)
Andrew	Programming	Marine Biology	Ballroom Dance
David	Constitutional Law	Basketweaving	Programming
Elizabeth	Marine Biology	Programming	Astronomy
Ethan	Basketweaving	Marine Biology	Astronomy
Katharine	Programming	Basketweaving	Marine Biology

Andrew's first and second choices are fully enrolled, but his third choice has openings. Andrew will be enrolled in Ballroom Dance.

David's first choice has openings. David will be enrolled in Constitutional Law.

All three of Elizabeth's choices are fully enrolled. Elizabeth will remain unassigned to an elective.

Ethan's first choice has one opening left. Ethan will be enrolled in Basketweaving. Note that Basketweaving is now fully enrolled.

All three of Katharine's choices are now fully enrolled. Katharine will remain unassigned to an elective.

In this problem, the school is modeled by the class `School`. Students and electives are modeled by the classes `Student` and `Elective` respectively.

The `School` class includes the following methods and private data:

- `studentList`—This `ArrayList` holds the list of students in the order in which the students should be scheduled.

- `electiveList`—This `ArrayList` holds the electives that students may choose.

- `getElectiveByName`—This method returns the `Elective` in `electiveList` with the given name.

- `assignElectivesToStudents`—This method encapsulates the functionality of assigning students (if possible) their first, second, or third elective choice.

- `studentsWithoutElectives`—This method returns an `ArrayList` containing students that have not been assigned an elective.

GO ON TO THE NEXT PAGE.

```
public class School
{
    private ArrayList<Student> studentList;
    // each entry is an instance of a
    // Student representing one student
    // at the school; students are in
    // the order they should be scheduled

    private ArrayList<Elective> electiveList;
    // each entry is an instance of an
    // Elective representing one elective
    // offered at the school

    // precondition: name is the name of an
    //     Elective in electiveList
    // postcondition: returns the Elective
    //     in electiveList with the given
    //     name
    private Elective getElectiveByName (String name)
    { /* to be implemented in part (a) */ }
        // postcondition: returns the size
    // of electiveList
    private int getElectiveListSize()
{
        return electiveList.size();
    }
    private int getStudentListSize()
{
        return studentList.size();
    }

    // postcondition: All Students in
    //     studentList have been either
    //     assigned their first available
    //     elective choice or not assigned;
    //     All Electives in electiveList have
    //     been updated appropriately as
    //     Students are assigned to them
    public void assignElectivesToStudents ( )
    { /* to be implemented in part (b) */ }

    // postcondition: returns a list of
    //     those Students who have not been
    //     assigned an Elective
    public ArrayList<Student>
       studentsWithoutElectives ( )
    { /* to be implemented in part (c) */}

    // ... constructors, other methods,
    //     and other private data not shown
}
```

The Student class includes the following methods and private data:

- getChoice—This method returns the name of the given elective choice of the student. The first elective choice has index 0, the second has index 1, and the third has index 2.

- hasElective—This method returns true if the student has been assigned an elective; it returns false otherwise.

- assignElective—This method assigns the given elective to this student.

```
public class Student
{
    // precondition: 0 <= index < 3
    // postcondition: returns the name
    //     of the given elective choice
    public String getChoice (int index)
    { /* code not shown */}

    // postcondition: returns true if
    //     an Elective has been assigned
    //     to this Student
    public boolean hasElective ( )
    { /* code not shown */ }

    // precondition: e is not null
    // postcondition: e has been assigned
    //     to this Student; e has not been
    //     modified
    public void assignElective (Elective e)
    { /* code not shown */ }

    // ... constructors, other methods,
    // and other private data not shown
}
```

The Elective class includes the following methods:

- getName—This method returns the name of this elective.

- getMaxClassSize—This method returns the maximum number of students that can be assigned to this elective.

- getClassSize—This method returns the number of students that have been assigned to this elective.

- addStudent—This method assigns the given student to this elective.

GO ON TO THE NEXT PAGE.

```
public class Elective
{
    // postcondition: returns the name
    //     of this Elective
    public String getName ( )
    { /* code not shown */ }

    // postcondition: returns the
    //     maximum number of Students
    //     that can be added to this
    //     Elective
    public int getMaxClassSize ( )
    { /* code not shown */ }

    // postcondition: returns the
    //     number of Students that have
    //     been added to this Elective;
    //     0 < = getClassSize ( ) < =
    //     getMaxClassSize ( )
    public int getClassSize ( )
    { /* code not shown */ }

    // precondition: getClassSize ( ) <
    //     getMaxClassSize ( ); s is not null
    // postcondition: s has been added to
    //     this Elective; getClassSize ( ) has
    //     been increased by 1
    public void addStudent (Student s)
    { /* code not shown */ }

    // ... constructors, other methods,
    //     and other private data not shown
}
```

(a) Write the `School` method `getElectiveByName`. Method `getElectiveByName` should return the `Elective` in `electiveList` that has the given name.

Complete method `getElectiveByName` below.

```
    // precondition: name is the name of an
    //     Elective in electiveList
    // postcondition: returns the Elective in
    //     electiveList with the given name
    private Elective getElectiveByName (String name)
```

(b) Write the `School` method `assignElectivesToStudents`. Method `assignElectivesToStudents` should assign electives to students as described at the beginning of this question.

In writing method `assignElectivesToStudents` you may use the `private` helper method `getElectiveBy-Name` specified in part (a). Assume that `getElectiveByName` works as specified, regardless of what you wrote in part (a). Solutions that reimplement functionality provided by this method, rather than invoking itt, will not receive full credit.

Complete method `assignElectivesToStudents` below

```
// postcondition: All Students in
//      studentList have been either
//      assigned their first available
//      elective choice or not assigned;
//      All electives in electiveList have
//      been updated appropriately as
//      Students are assigned to them
public void assignElectivesToStudents ( )
```

(c) Write the `School` method `studentsWithoutElectives`. Method `studentsWithoutElectives` should return `ArrayList` of all Students in `studentList` who do not have an `Elective` assigned to them.

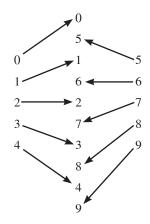Complete method `studentsWithoutElectives` below

```
// postcondition: returns a list of those
//       Students who have not been assigned
//       an Elective
public ArrayList studentsWithoutElectives ( )
```

2. Consider a deck of n cards where n is even and each card is uniquely labeled from 1 to n.

   A *shuffle* is performed when the deck is divided into two stacks and the stacks are interlaced so that a new stack is formed by alternately taking cards from each stack.

   For instance, a deck of ten cards is in order when the card labeled 0 is on the top of the deck and the card labeled 9 is on the bottom of the deck.

   Dividing the deck in half produces two stacks of cards – one stack with cards 0 through 4, the other with cards 5 through 9. Interlacing the stacks produces a deck in the following order:

   The number of times needed to shuffle the deck until it returns to its original order is called the *reorder count*. Note that the reorder count for a deck of ten cards is six:

Shuffle number

Original	1	2	3	4	5	6
0	0	0	0	0	0	0
1	5	7	8	4	2	1
2	1	5	7	8	4	2
3	6	3	6	3	6	3
4	2	1	5	7	8	4
5	7	8	4	2	1	5
6	3	6	3	6	3	6
7	8	4	2	1	5	7
8	4	2	1	5	7	8
9	9	9	9	9	9	9

GO ON TO THE NEXT PAGE.

A deck is modeled by the following incomplete declaration of the **Deck** class:

```
public class Deck
{
    private int [ ] cards;

    public Deck (int numCards)
    { /* code not shown */ }

    public boolean inOrder ( )
    { /* to be implemented in part (a) */ }

    public void shuffle ( )
    { /* to be implemented in part (b) */ }

    public int reorderingCount ( )
    { /* to be implemented in part (c) */ }
}
```

(a) Write the Deck method inOrder. Method inOrder should return true if the cards in the deck are in numerical order from 0 to cards.length – 1 and should return false otherwise. Cards are in numerical order if cards [k] = = k for all 0 < = k < cards. length.

Complete method inOrder below.

```
    // precondition: For all k such that
    //     0 < = k < cards.length,
    //     0 < = cards [k] < cards.length and
    //     each cards [k] is unique
    // postcondition: returns true if
    //     cards [k] = = k for all
    //     0 < = k < cards.length; returns
    //     false otherwise
    public boolean inOrder ( )
```

GO ON TO THE NEXT PAGE.

(b) Write the Deck method shuffle. Method shuffle should divide the deck into two equal stacks and interlace them evenly as described at the beginning of this question.

Complete method shuffle below.

```
// postcondition: the deck is shuffled by
//        dividing the deck into two equal stacks
//        that are evenly interlaced
public void shuffle ( )
```

(c) Write the Deck method reorderCount. Method reorderCount should return the number of shuffles necessary to return the deck to its original order.

In writing method reorderCount you may use the methods inOrder and shuffle as specified in parts (a) and (b). Assume that inOrder and shuffle work as specified, regardless of what you wrote in parts (a) and (b). Solutions that reimplement functionality provided by these methods, rather than invoking them, will not receive full credit.

Complete method reorderCount below.

```
// postcondition: returns the number of
//      shuffles necessary to return the cards
//      in the deck to their original numerical
//      order such that inOrder ( ) = = true; the
//      cards in the deck are in their original
//      numerical order
public int reorderCount ( )
```

GO ON TO THE NEXT PAGE.

3. Consider the design of an electronic cookbook modeled with the following class declarations:

```
public class Cookbook
{
    private ArrayList recipeList;
    // each entry is an instance of a
    // Recipe representing one recipe
    // in the cookbook
    // precondition: numPeople > 0
    // postcondition: All recipes in
    //     recipeList have been converted to
    //     serve numPeople number of people

    public void standardize (int numPeople)
    { /* code not shown */ }

    // ... constructors, other methods,
    //     and other private data not shown
}

public class Ingredient
{
    private String name;
    // the name of this ingredient

    private double amount;
    // the amount of this ingredient needed
    // in the recipe
    // postcondition: returns the amount of
    //     this ingredient needed in the
    //     recipe

    public double getAmount ( )
    { /* code not shown */ }
    // precondition: amt > 0.0
    // postcondition: amount has been set
    //     to amt

    public void setAmount (double amt)
    { /* code not shown */ }
    // precondition: newNumber > 0
    // postcondition: numberServed has
    // been set to newNumber

    public void setNumberServed(int newNumber)
    { /* code not shown */ }
    // ... constructors and other methods
    //     not shown
}
```

GO ON TO THE NEXT PAGE.

(a) A recipe in the cookbook is modeled by the class Recipe with the following data and operations:

Data

- return the list of ingredients
- the name of the recipe
- the list of ingredients used in the recipe
- the description of the preparation process for the recipe
- the number of people served by the recipe

Operations

- create a recipe with a given name and number of people served
- add an ingredient to the recipe
- set the description of the preparation process for the recipe
- return the name of the recipe
- return the number of people served by the recipe
- scale the recipe to serve a given new number of people by changing the amount of each ingredient appropriately

Write the definition of the class Recipe, showing the appropriate data definitions and constructor and method signatures. You should *not* write the implementations of the constructor or methods for the Recipe class.

(b) Using the signature you wrote in part (a), write the implementation for the method that scales the recipe to serve a given new number of people.

In writing this method, you may call any of the methods in the Recipe class (as you defined it in part (a)) or in the Ingredient class. Assume that these methods work as specified.

(c) Write the Cookbook method standardize as described at the beginning of the question.

In writing this method, you may call any of the methods in the Recipe class (as you defined it in part (a)). Assume that these methods work as specified.

Complete method standardize below.

```
// precondition: numPeople > 0
// postcondition: All recipes in
//      recipeList have been scaled to
//      serve numPeople number of people
public void standardize (int numPeople)
```

GO ON TO THE NEXT PAGE.

4. Consider a school that contains x number of students that all start their first period class in one of n classrooms. This scenario can be represented using three classes. The `School` class contains an array of all the `Classrooms` in the school. The `Classroom` class has fields for the teacher in charge of the room `teacherName` and an array of all the `Students` in the classroom `Students`. The `Student` class has a field for the name of the student `studentName` and the ID number of the student `studentID`.

The `School` class contains a method `findStudent` that takes a teacher's name and a student ID as arguments and returns the name of the student. The method utilizes a sequential search algorithm to find the correct classroom and a binary search algorithm to find the correct student. If the student is not found in the school, the method returns "`Student Not Found.`"

Write the complete `School`, `Classroom`, and `Student` classes, including any instance variables, constructors, and necessary methods. You may assume that the student ID numbers in each classroom are sorted in ascending order.

STOP

END OF EXAM

The Princeton Review®

1.

YOUR NAME: _____
(Print)   Last   First   M.I.

SIGNATURE: _____ DATE: _____ / _____ / _____

HOME ADDRESS: _____
(Print)   Number and Street

_____
City   State   Zip Code

PHONE NO.: _____

IMPORTANT: Please fill in these boxes exactly as shown on the back cover of your test book.

2. TEST FORM

_____

3. TEST CODE

0	Ⓐ	Ⓙ	0
1	Ⓑ	Ⓚ	1
2	Ⓒ	Ⓛ	2
3	Ⓓ	Ⓜ	3
4	Ⓔ	Ⓝ	4
5	Ⓕ	Ⓞ	5
6	Ⓖ	Ⓟ	6
7	Ⓗ	Ⓠ	7
8	Ⓘ	Ⓡ	8
9			9

4. REGISTRATION NUMBER

0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9

6. DATE OF BIRTH

Month	Day		Year	
◯ JAN				
◯ FEB	0	0	0	0
◯ MAR	1	1	1	1
◯ APR	2	2	2	2
◯ MAY	3	3	3	3
◯ JUN		4	4	4
◯ JUL		5	5	5
◯ AUG		6	6	6
◯ SEP		7	7	7
◯ OCT		8	8	8
◯ NOV		9	9	9
◯ DEC				

7. GENDER

◯ MALE
◯ FEMALE

The **Princeton** Review®

5. YOUR NAME

First 4 letters of last name				FIRST INIT	MID INIT
Ⓐ	Ⓐ	Ⓐ	Ⓐ	Ⓐ	Ⓐ
Ⓑ	Ⓑ	Ⓑ	Ⓑ	Ⓑ	Ⓑ
Ⓒ	Ⓒ	Ⓒ	Ⓒ	Ⓒ	Ⓒ
Ⓓ	Ⓓ	Ⓓ	Ⓓ	Ⓓ	Ⓓ
Ⓔ	Ⓔ	Ⓔ	Ⓔ	Ⓔ	Ⓔ
Ⓕ	Ⓕ	Ⓕ	Ⓕ	Ⓕ	Ⓕ
Ⓖ	Ⓖ	Ⓖ	Ⓖ	Ⓖ	Ⓖ
Ⓗ	Ⓗ	Ⓗ	Ⓗ	Ⓗ	Ⓗ
Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ	Ⓘ
Ⓙ	Ⓙ	Ⓙ	Ⓙ	Ⓙ	Ⓙ
Ⓚ	Ⓚ	Ⓚ	Ⓚ	Ⓚ	Ⓚ
Ⓛ	Ⓛ	Ⓛ	Ⓛ	Ⓛ	Ⓛ
Ⓜ	Ⓜ	Ⓜ	Ⓜ	Ⓜ	Ⓜ
Ⓝ	Ⓝ	Ⓝ	Ⓝ	Ⓝ	Ⓝ
Ⓞ	Ⓞ	Ⓞ	Ⓞ	Ⓞ	Ⓞ
Ⓟ	Ⓟ	Ⓟ	Ⓟ	Ⓟ	Ⓟ
Ⓠ	Ⓠ	Ⓠ	Ⓠ	Ⓠ	Ⓠ
Ⓡ	Ⓡ	Ⓡ	Ⓡ	Ⓡ	Ⓡ
Ⓢ	Ⓢ	Ⓢ	Ⓢ	Ⓢ	Ⓢ
Ⓣ	Ⓣ	Ⓣ	Ⓣ	Ⓣ	Ⓣ
Ⓤ	Ⓤ	Ⓤ	Ⓤ	Ⓤ	Ⓤ
Ⓥ	Ⓥ	Ⓥ	Ⓥ	Ⓥ	Ⓥ
Ⓦ	Ⓦ	Ⓦ	Ⓦ	Ⓦ	Ⓦ
Ⓧ	Ⓧ	Ⓧ	Ⓧ	Ⓧ	Ⓧ
Ⓨ	Ⓨ	Ⓨ	Ⓨ	Ⓨ	Ⓨ
Ⓩ	Ⓩ	Ⓩ	Ⓩ	Ⓩ	Ⓩ

1. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
2. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
3. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
4. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
5. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
6. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
7. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
8. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
9. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
10. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

11. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
12. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
13. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
14. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
15. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
16. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
17. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
18. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
19. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
20. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

21. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
22. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
23. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
24. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
25. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
26. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
27. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
28. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
29. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
30. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

31. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
32. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
33. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
34. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
35. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
36. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
37. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
38. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
39. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ
40. Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ