MULTIPLE-CHOICE QUESTIONS ON ARRAYS AND ARRAY LISTS

1. Which of the following correctly initializes an array `arr` to contain four elements each with value `0`?

   I
   ```
   int[] arr = {0, 0, 0, 0};
   ```

   II
   ```
   int[] arr = new int[4];
   ```

   III
   ```
   int[] arr = new int[4];
   for (int i = 0; i < arr.length; i++)
       arr[i] = 0;
   ```

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

2. The following program segment is intended to find the index of the first negative integer in `arr[0]` ... `arr[N-1]`, where `arr` is an array of `N` integers.

   ```
   int i = 0;
   while (arr[i] >= 0)
   {
       i++;
   }
   location = i;
   ```

   This segment will work as intended
   (A) always.
   (B) never.
   (C) whenever `arr` contains at least one negative integer.
   (D) whenever `arr` contains at least one nonnegative integer.
   (E) whenever `arr` contains no negative integers.

3. Refer to the following code segment. You may assume that `arr` is an array of `int` values.

   ```
   int sum = arr[0], i = 0;
   while (i < arr.length)
   {
       i++;
       sum += arr[i];
   }
   ```

   Which of the following will be the result of executing the segment?
   (A) Sum of `arr[0]`, `arr[1]`, ..., `arr[arr.length-1]` will be stored in `sum`.
   (B) Sum of `arr[1]`, `arr[2]`, ..., `arr[arr.length-1]` will be stored in `sum`.
   (C) Sum of `arr[0]`, `arr[1]`, ..., `arr[arr.length]` will be stored in `sum`.
   (D) An infinite loop will occur.
   (E) A run-time error will occur.

4. Refer to the following code segment. You may assume that array `arr1` contains elements `arr1[0]`, `arr1[1]`, ..., `arr1[N-1]`, where `N = arr1.length`.

```
int count = 0;
for (int i = 0; i < N; i++)
    if (arr1[i] != 0)
    {
        arr1[count] = arr1[i];
        count++;
    }
int[] arr2 = new int[count];
for (int i = 0; i < count; i++)
    arr2[i] = arr1[i];
```

If array `arr1` initially contains the elements 0, 6, 0, 4, 0, 0, 2 in this order, what will `arr2` contain after execution of the code segment?

(A) 6, 4, 2
(B) 0, 0, 0, 0, 6, 4, 2
(C) 6, 4, 2, 4, 0, 0, 2
(D) 0, 6, 0, 4, 0, 0, 2
(E) 6, 4, 2, 0, 0, 0, 0

5. Consider this program segment:

```
for (int i = 2; i <= k; i++)
    if (arr[i] < someValue)
        System.out.print("SMALL");
```

What is the maximum number of times that `SMALL` can be printed?

(A) `0`
(B) `1`
(C) `k - 1`
(D) `k - 2`
(E) `k`

6. What will be output from the following code segment, assuming it is in the same class as the `doSomething` method?

```
int[] arr = {1, 2, 3, 4};
doSomething(arr);
System.out.print(arr[1] + " ");
System.out.print(arr[3]);
    ...
public void doSomething(int[] list)
{
    int[] b = list;
    for (int i = 0; i < b.length; i++)
        b[i] = i;
}
```

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

7. Consider writing a program that reads the lines of any text file into a sequential list of lines. Which of the following is a good reason to implement the list with an `ArrayList` of `String` objects rather than an array of `String` objects?
   (A) The `get` and `set` methods of `ArrayList` are more convenient than the `[]` notation for arrays.
   (B) The `size` method of `ArrayList` provides instant access to the length of the list.
   (C) An `ArrayList` can contain objects of any type, which leads to greater generality.
   (D) If any particular text file is unexpectedly long, the `ArrayList` will automatically be resized. The array, by contrast, may go out of bounds.
   (E) The `String` methods are easier to use with an `ArrayList` than with an array.

8. Consider writing a program that produces statistics for long lists of numerical data. Which of the following is the best reason to implement each list with an array of `int` (or `double`), rather than an `ArrayList` of `Integer` (or `Double`) objects?
   (A) An array of primitive number types is more efficient to manipulate than an `ArrayList` of wrapper objects that contain numbers.
   (B) Insertion of new elements into a list is easier to code for an array than for an `ArrayList`.
   (C) Removal of elements from a list is easier to code for an array than for an `ArrayList`.
   (D) Accessing individual elements in the middle of a list is easier for an array than for an `ArrayList`.
   (E) Accessing all the elements is more efficient in an array than in an `ArrayList`.

Refer to the following classes for Questions 9–12.

```
public class Address
{
    private String name;
    private String street;
    private String city;
    private String state;
    private String zip;

    //constructors
        ...

    //accessors
    public String getName()
    { return name; }
    public String getStreet()
    { return street; }
    public String getCity()
    { return city; }
    public String getState()
    { return state; }
    public String getZip()
    { return zip; }
}

public class Student
{
    private int idNum;
    private double gpa;
    private Address address;

    //constructors
        ...

    //accessors
    public Address getAddress()
    { return address; }
    public int getIdNum()
    { return idNum; }
    public double getGpa()
    { return gpa; }
}
```

9. A client method has this declaration, followed by code to initialize the list:

```
Address[] list = new Address[100];
```

Here is a code segment to generate a list of *names only*.

```
for (Address a : list)
    /* line of code */
```

Which is a correct `/* line of code */`?

(A) `System.out.println(Address[i].getName());`
(B) `System.out.println(list[i].getName());`
(C) `System.out.println(a[i].getName());`
(D) `System.out.println(a.getName());`
(E) `System.out.println(list.getName());`

10. The following code segment is to print out a list of addresses:

```
for (Address addr : list)
{
    /* more code */
}
```

Which is a correct replacement for `/* more code */`?

I
```
System.out.println(list[i].getName());
System.out.println(list[i].getStreet());
System.out.print(list[i].getCity() + ", ");
System.out.print(list[i].getState() + " ");
System.out.println(list[i].getZip());
```

II
```
System.out.println(addr.getName());
System.out.println(addr.getStreet());
System.out.print(addr.getCity() + ", ");
System.out.print(addr.getState() + " ");
System.out.println(addr.getZip());
```

III
```
System.out.println(addr);
```

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

11. A client method has this declaration:

```
Student[] allStudents = new Student[NUM_STUDS];  //NUM_STUDS is
                                                 //an int constant
```

Here is a code segment to generate a list of `Student` names only. (You may assume that `allStudents` has been initialized.)

```
for (Student student : allStudents)
    /* code to print list of names */
```

Which is a correct replacement for `/* code to print list of names */`?

(A) `System.out.println(allStudents.getName());`
(B) `System.out.println(student.getName());`
(C) `System.out.println(student.getAddress().getName());`
(D) `System.out.println(allStudents.getAddress().getName());`
(E) `System.out.println(student[i].getAddress().getName());`

12. Here is a method that locates the `Student` with the highest `idNum`:

```
/** Precondition: Array stuArr of Student is initialized.
 *  @return Student with highest idNum
 */
public static Student locate(Student[] stuArr)
{
    /* method body */
}
```

Which of the following could replace `/* method body */` so that the method works as intended?

I
```
int max = stuArr[0].getIdNum();
for (Student student : stuArr)
    if (student.getIdNum() > max)
    {
        max = student.getIdNum();
        return student;
    }
return stuArr[0];
```

II
```
Student highestSoFar = stuArr[0];
int max = stuArr[0].getIdNum();
for (Student student : stuArr)
    if(student.getIdNum() > max)
    {
        max = student.getIdNum();
        highestSoFar = student;
    }
return highestSoFar;
```

III
```
int maxPos = 0;
for(int i = 1; i < stuArr.length; i++)
    if(stuArr[i].getIdNum() > stuArr[maxPos].getIdNum())
        maxPos = i;
return stuArr[maxPos];
```

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

Questions 13–15 refer to the `Ticket` and `Transaction` classes below.

```
public class Ticket
{
    private String row;
    private int seat;
    private double price;

    //constructor
    public Ticket(String aRow, int aSeat, double aPrice)
    {
        row = aRow;
        seat = aSeat;
        price = aPrice;
    }

    //accessors getRow(), getSeat(), and getPrice()
        ...
}

public class Transaction
{
    private int numTickets;
    private Ticket[] tickList;

    //constructor
    public Transaction(int numTicks)
    {
        numTickets = numTicks;
        tickList = new Ticket[numTicks];
        String theRow;
        int theSeat;
        double thePrice;
        for (int i = 0; i < numTicks; i++)
        {
            < read user input for theRow, theSeat, and thePrice >
                ...

            /* more code */
        }
    }

    /** @return total amount paid for this transaction */
    public double totalPaid()
    {
        double total = 0.0;
        /* code to calculate amount */
        return total;
    }
}
```

13. Which of the following correctly replaces `/* more code */` in the `Transaction` constructor to initialize the `tickList` array?

    (A) `tickList[i] = new Ticket(getRow(), getSeat(), getPrice());`

    (B) `tickList[i] = new Ticket(theRow, theSeat, thePrice);`

    (C) `tickList[i] = new tickList(getRow(), getSeat(), getPrice());`

    (D) `tickList[i] = new tickList(theRow, theSeat, thePrice);`

    (E) `tickList[i] = new tickList(numTicks);`

14. Which represents correct `/* code to calculate amount */` in the `totalPaid` method?

    (A)
    ```
    for (Ticket t : tickList)
        total += t.price;
    ```

    (B)
    ```
    for (Ticket t : tickList)
        total += tickList.getPrice();
    ```

    (C)
    ```
    for (Ticket t : tickList)
        total += t.getPrice();
    ```

    (D)
    ```
    Transaction T;
    for (Ticket t : T)
        total += t.getPrice();
    ```

    (E)
    ```
    Transaction T;
    for (Ticket t : T)
        total += t.price;
    ```

15. Suppose it is necessary to keep a list of all ticket transactions. Assuming that there are `NUMSALES` transactions, a suitable declaration would be

    (A) `Transaction[] listOfSales = new Transaction[NUMSALES];`

    (B) `Transaction[] listOfSales = new Ticket[NUMSALES];`

    (C) `Ticket[] listOfSales = new Transaction[NUMSALES];`

    (D) `Ticket[] listOfSales = new Ticket[NUMSALES];`

    (E) `Transaction[] Ticket = new listOfSales[NUMSALES];`

16. The following code fragment is intended to find the smallest value in `arr[0]...arr[n-1]`.

```
/** Precondition:
 *   - arr is an array, arr.length = n.
 *   - arr[0]...arr[n-1] initialized with integers.
 *  Postcondition: min = smallest value in arr[0]...arr[n-1].
 */
int min = arr[0];
int i = 1;
while (i < n)
{
    i++;
    if (arr[i] < min)
        min = arr[i];
}
```

This code is incorrect. For the segment to work as intended, which of the following modifications could be made?

I Change the line

```
int i = 1;
```

to

```
int i = 0;
```

Make no other changes.

II Change the body of the `while` loop to

```
{
    if (arr[i] < min)
        min = arr[i];
    i++;
}
```

Make no other changes.

III Change the test for the `while` loop as follows:

```
while (i <= n)
```

Make no other changes.

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

17. Refer to method `match` below:

```
/** @param v an array of int sorted in increasing order
 *  @param w an array of int sorted in increasing order
 *  @param N the number of elements in array v
 *  @param M the number of elements in array w
 *  @return true if there is an integer k that occurs
 *  in both arrays; otherwise returns false
 *  Precondition:
 *    v[0]..v[N-1] and w[0]..w[M-1] initialized with integers.
 *    v[0] < v[1] < .. < v[N-1] and w[0] < w[1] < .. < w[M-1].
 */
public static boolean match(int[] v, int[] w, int N, int M)
{
    int vIndex = 0, wIndex = 0;
    while (vIndex < N && wIndex < M)
    {
        if (v[vIndex] == w[wIndex])
            return true;
        else if (v[vIndex] < w[wIndex])
            vIndex++;
        else
            wIndex++;
    }
    return false;
}
```

Assuming that the method has not been exited, which assertion is true at the end of every execution of the `while` loop?

(A) `v[0]..v[vIndex-1]` and `w[0]..w[wIndex-1]` contain no common value, `vIndex` $\leq$ `N` and `wIndex` $\leq$ `M`.

(B) `v[0]..v[vIndex]` and `w[0]..w[wIndex]` contain no common value, `vIndex` $\leq$ `N` and `wIndex` $\leq$ `M`.

(C) `v[0]..v[vIndex-1]` and `w[0]..w[wIndex-1]` contain no common value, `vIndex` $\leq$ `N-1` and `wIndex` $\leq$ `M-1`.

(D) `v[0]..v[vIndex]` and `w[0]..w[wIndex]` contain no common value, `vIndex` $\leq$ `N-1` and `wIndex` $\leq$ `M-1`.

(E) `v[0]..v[N-1]` and `w[0]..w[M-1]` contain no common value, `vIndex` $\leq$ `N` and `wIndex` $\leq$ `M`.

18. Consider this class:

```
public class Book
{
    private String title;
    private String author;
    private boolean checkoutStatus;

    public Book(String bookTitle, String bookAuthor)
    {
        title = bookTitle;
        author = bookAuthor;
        checkoutStatus = false;
    }

    /** Change checkout status. */
    public void changeStatus()
    { checkoutStatus = !checkoutStatus; }

    //Other methods are not shown.
}
```

A client program has this declaration:

```
Book[] bookList = new Book[SOME_NUMBER];
```

Suppose `bookList` is initialized so that each `Book` in the list has a title, author, and checkout status. The following piece of code is written, whose intent is to change the checkout status of each book in `bookList`.

```
for (Book b : bookList)
    b.changeStatus();
```

Which is *true* about this code?

(A) The `bookList` array will remain unchanged after execution.
(B) Each book in the `bookList` array will have its checkout status changed, as intended.
(C) A `NullPointerException` may occur.
(D) A run-time error will occur because it is not possible to modify objects using the for-each loop.
(E) A logic error will occur because it is not possible to modify objects in an array without accessing the indexes of the objects.

Consider this class for Questions 19 and 20:

```
public class BingoCard
{
    private int[] card;

    /** Default constructor: Creates BingoCard with
     *  20 random digits in the range 1 - 90.
     */
    public BingoCard()
    { /* implementation not shown */ }

    /* Display BingoCard. */
    public void display()
    { /* implementation not shown */ }
        ...
}
```

A program that simulates a bingo game declares an array of `BingoCard`. The array has `NUMPLAYERS` elements, where each element represents the card of a different player. Here is a code segment that creates all the bingo cards in the game:

```
/* declare array of BingoCard */
/* construct each BingoCard */
```

19. Which of the following is a correct replacement for

    `/* declare array of BingoCard */`?

    (A) `int[] BingoCard = new BingoCard[NUMPLAYERS];`

    (B) `BingoCard[] players = new int[NUMPLAYERS];`

    (C) `BingoCard[] players = new BingoCard[20];`

    (D) `BingoCard[] players = new BingoCard[NUMPLAYERS];`

    (E) `int[] players = new BingoCard[NUMPLAYERS];`

20. Assuming that `players` has been declared as an array of `BingoCard`, which of the following is a correct replacement for

    `/* construct each BingoCard */`

    I
    ```
    for (BingoCard card : players)
        card = new BingoCard();
    ```

    II
    ```
    for (BingoCard card : players)
        players[card] = new BingoCard();
    ```

    III
    ```
    for (int i = 0; i < players.length; i++)
        players[i] = new BingoCard();
    ```

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

21. Which declaration will cause an error?

I `List<String> stringList = new ArrayList<String>();`

II `List<int> intList = new ArrayList<int>();`

III `ArrayList<String> compList = new ArrayList<String>();`

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

22. Consider these declarations:

```
List<String> strList = new ArrayList<String>();
String ch = " ";
Integer intOb = new Integer(5);
```

Which statement will cause an error?
(A) `strList.add(ch);`
(B) `strList.add(new String("handy andy"));`
(C) `strList.add(intOb.toString());`
(D) `strList.add(ch + 8);`
(E) `strList.add(intOb + 8);`

23. Let `list` be an `ArrayList<Integer>` containing these elements:

```
2 5 7 6 0 1
```

Which of the following statements would *not* cause an error to occur? Assume that each statement applies to the given list, independent of the other statements.
(A) `Object ob = list.get(6);`
(B) `Integer intOb = list.add(3.4);`
(C) `list.add(6, 9);`
(D) `Object x = list.remove(6);`
(E) `Object y = list.set(6, 8);`

24. Refer to method `insert` below:

```
/** @param list an ArrayList of String objects
 *  @param element a String object
 *  Precondition: list contains String values sorted
 *                in decreasing order.
 *  Postcondition: element inserted in its correct position in list.
 */
public void insert(List<String> list, String element)
{
    int index = 0;
    while (element.compareTo(list.get(index)) < 0)
        index++;
    list.add(index, element);
}
```

Assuming that the type of `element` is compatible with the objects in the list, which is a *true* statement about the `insert` method?

(A) It works as intended for all values of `element`.
(B) It fails for all values of `element`.
(C) It fails if `element` is greater than the first item in `list` and works in all other cases.
(D) It fails if `element` is smaller than the last item in `list` and works in all other cases.
(E) It fails if `element` is either greater than the first item or smaller than the last item in `list` and works in all other cases.

25. Consider the following code segment, applied to `list`, an `ArrayList` of `Integer` values.

```
int len = list.size();
for (int i = 0; i < len; i++)
{
    list.add(i + 1, new Integer(i));
    Object x = list.set(i, new Integer(i + 2));
}
```

If `list` is initially `6 1 8`, what will it be following execution of the code segment?

(A) `2 3 4 2 1 8`
(B) `2 3 4 6 2 2 0 1 8`
(C) `2 3 4 0 1 2`
(D) `2 3 4 6 1 8`
(E) `2 3 3 2`

Questions 26 and 27 are based on the `Coin` and `Purse` classes given below:

```
/* A simple coin class */
public class Coin
{
    private double value;
    private String name;

    //constructor
    public Coin(double coinValue, String coinName)
    {
        value = coinValue;
        name = coinName;
    }

    /** @return the value of this coin */
    public double getValue()
    { return value; }

    /** @return the name of this coin */
    public String getName()
    { return name; }

    /** @param obj a Coin object
     *  @return true if this coin equals obj; otherwise false
     */
    public boolean equals(Object obj)
    { return name.equals(((Coin) obj).name); }

    //Other methods are not shown.
}

/* A purse holds a collection of coins */
public class Purse
{
    private List<Coin> coins;

    /** Creates an empty purse. */
    public Purse()
    { coins = new ArrayList<Coin>(); }

    /** Adds aCoin to the purse.
     *  @param aCoin the coin to be added to the purse
     */
    public void add(Coin aCoin)
    { coins.add(aCoin); }

    /** @return the total value of coins in purse */
    public double getTotal()
    { /* implementation not shown */}

}
```

26. Here is the `getTotal` method from the `Purse` class:

```
/** @return the total value of coins in purse */
public double getTotal()
{
    double total = 0;
    /* more code */
    return total;
}
```

Which of the following is a correct replacement for `/* more code */`?

(A)
```
for (Coin c : coins)
{
    c = coins.get(i);
    total += c.getValue();
}
```

(B)
```
for (Coin c : coins)
{
    Coin value = c.getValue();
    total += value;
}
```

(C)
```
for (Coin c : coins)
{
    Coin c = coins.get(i);
    total += c.getValue();
}
```

(D)
```
for (Coin c : coins)
{
    total += coins.getValue();
}
```

(E)
```
for (Coin c : coins)
{
    total += c.getValue();
}
```

27. Two coins are said to *match* each other if they have the same name or the same value. You may assume that coins with the same name have the same value and coins with the same value have the same name. A boolean method `find` is added to the `Purse` class:

```
/** @return true if the purse has a coin that matches aCoin,
 *  false otherwise
 */
public boolean find(Coin aCoin)
{
    for (Coin c : coins)
    {
        /* code to find match */
    }
    return false;
}
```

Which is a correct replacement for `/* code to find match */`?

I
```
if (c.equals(aCoin))
    return true;
```

II
```
if ((c.getName()).equals(aCoin.getName()))
    return true;
```

III
```
if ((c.getValue()).equals(aCoin.getValue()))
    return true;
```

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

28. Which of the following initializes an 8×10 matrix with integer values that are perfect squares? (0 is a perfect square.)

I
```
int[][] mat = new int[8][10];
```

II
```
int[][] mat = new int[8][10];
for (int r = 0; r < mat.length; r++)
    for (int c = 0; c < mat[r].length; c++)
        mat[r][c] = r * r;
```

III
```
int[][] mat = new int[8][10];
for (int c = 0; c < mat[r].length; c++)
    for (int r = 0; r < mat.length; r++)
        mat[r][c] = c * c;
```

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

29. Consider a class that has this private instance variable:

```
private int[][] mat;
```

The class has the following method, `alter`.

```
public void alter(int c)
{
    for (int i = 0; i < mat.length; i++)
        for (int j = c + 1; j < mat[0].length; j++)
            mat[i][j-1] = mat[i][j];
}
```

If a 3 × 4 matrix `mat` is

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

then `alter(1)` will change `mat` to

(A)
```
1 5 7 7
2 6 8 8
3 7 9 9
```

(B)
```
1 5 7
2 6 8
3 7 9
```

(C)
```
1 3 5 7
3 5 7 9
```

(D)
```
1 3 5 7
3 5 7 9
3 5 7 9
```

(E)
```
1 7 7 7
2 8 8 8
3 9 9 9
```

30. Consider the following method that will alter the matrix `mat`:

```
/** @param mat the initialized matrix
 *  @param row the row number
 */
public static void matStuff(int[][] mat, int row)
{
    int numCols = mat[0].length;
    for (int col = 0; col < numCols; col++)
        mat[row][col] = row;
}
```

Suppose `mat` is originally

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

After the method call `matStuff(mat,2)`, matrix `mat` will be

(A)
```
1  4  9  0
2  7  8  6
2  2  2  2
```

(B)
```
1  4  9  0
2  2  2  2
5  1  4  3
```

(C)
```
2  2  2  2
2  2  2  2
2  2  2  2
```

(D)
```
1  4  2  0
2  7  2  6
5  1  2  3
```

(E)
```
1  2  9  0
2  2  8  6
5  2  4  3
```

31. Assume that a square matrix `mat` is defined by

```
int[][] mat = new int[SIZE][SIZE];
//SIZE is an integer constant >= 2
```

What does the following code segment do?

```
for (int i = 0; i < SIZE - 1; i++)
    for (int j = 0; j < SIZE - i - 1; j++)
        swap(mat, i, j, SIZE - j - 1, SIZE - i - 1);
```

You may assume the existence of this `swap` method:

```
/** Interchange mat[a][b] and mat[c][d]. */
public void swap(int[][] mat, int a, int b, int c, int d)
```

(A) Reflects `mat` through its major diagonal. For example,

$$\begin{matrix} 2 & 6 \\ & \\ 4 & 3 \end{matrix} \longrightarrow \begin{matrix} 2 & 4 \\ & \\ 6 & 3 \end{matrix}$$

(B) Reflects `mat` through its minor diagonal. For example,

$$\begin{matrix} 2 & 6 \\ & \\ 4 & 3 \end{matrix} \longrightarrow \begin{matrix} 3 & 6 \\ & \\ 4 & 2 \end{matrix}$$

(C) Reflects `mat` through a horizontal line of symmetry. For example,

$$\begin{matrix} 2 & 6 \\ & \\ 4 & 3 \end{matrix} \longrightarrow \begin{matrix} 4 & 3 \\ & \\ 2 & 6 \end{matrix}$$

(D) Reflects `mat` through a vertical line of symmetry. For example,

$$\begin{matrix} 2 & 6 \\ & \\ 4 & 3 \end{matrix} \longrightarrow \begin{matrix} 6 & 2 \\ & \\ 3 & 4 \end{matrix}$$

(E) Leaves `mat` unchanged.

32. Consider a class `MatrixStuff` that has a private instance variable:

```
private int[][] mat;
```

Refer to method `alter` below that occurs in the `MatrixStuff` class. (The lines are numbered for reference.)

```
Line 1:  /** @param mat the matrix initialized with integers
Line 2:   *  @param c the column to be removed
Line 3:   *  Postcondition:
Line 4:   *   - Column c has been removed.
Line 5:   *   - The last column is filled with zeros.
Line 6:   */
Line 7:  public void alter(int[][] mat, int c)
Line 8:  {
Line 9:      for (int i = 0; i < mat.length; i++)
Line 10:         for (int j = c; j < mat[0].length; j++)
Line 11:             mat[i][j] = mat[i][j+1];
Line 12:     //code to insert zeros in rightmost column
Line 13:         ...
Line 14: }
```

The intent of the method `alter` is to remove column `c`. Thus, if the input matrix `mat` is

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

the method call `alter(mat, 1)` should change `mat` to

2	8	9	0
1	4	3	0
0	3	2	0

The method does not work as intended. Which of the following changes will correct the problem?

I Change line 10 to

```
for (int j = c; j < mat[0].length - 1; j++)
```

and make no other changes.

II Change lines 10 and 11 to

```
for (int j = c + 1; j < mat[0].length; j++)
    mat[i][j-1] = mat[i][j];
```

and make no other changes.

III Change lines 10 and 11 to

```
for (int j = mat[0].length - 1; j > c; j--)
    mat[i][j-1] = mat[i][j];
```

and make no other changes.

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

33. This question refers to the following method:

```
public static boolean isThere(String[][] mat, int row, int col,
    String symbol)
{
    boolean yes;
    int i, count = 0;
    for (i = 0; i < SIZE; i++)
        if (mat[i][col].equals(symbol))
            count++;
    yes = (count == SIZE);
    count = 0;
    for (i = 0; i < SIZE; i++)
        if (mat[row][i].equals(symbol))
            count++;
    return (yes || count == SIZE);
}
```

Now consider this code segment:

```
public final int SIZE = 8;
String[][] mat = new String[SIZE][SIZE];
```

Which of the following conditions on a matrix `mat` of the type declared in the code segment will by itself guarantee that

```
isThere(mat, 2, 2, "$")
```

will have the value `true` when evaluated?

I The element in row 2 and column 2 is `"$"`
II All elements in both diagonals are `"$"`
III All elements in column 2 are `"$"`

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

34. The method `changeNegs` below should replace every occurrence of a negative integer in its matrix parameter with `0`.

```
/** @param mat the matrix
 *  Precondition: mat is initialized with integers.
 *  Postcondition: All negative values in mat replaced with 0.
 */
public static void changeNegs(int[][] mat)
{
    /* code */
}
```

Which is correct replacement for `/* code */`?

I
```
for (int r = 0; r < mat.length; r++)
    for (int c = 0; c < mat[r].length; c++)
        if (mat[r][c] < 0)
            mat[r][c] = 0;
```

II
```
for (int c = 0; c < mat[0].length; c++)
    for (int r = 0; r < mat.length; r++)
        if (mat[r][c] < 0)
            mat[r][c] = 0;
```

III
```
for (int[] row : mat)
    for (int element : row)
        if (element < 0)
            element = 0;
```

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

35. A two-dimensional array of `double`, `rainfall`, will be used to represent the daily rainfall for a given year. In this scheme, `rainfall[month][day]` represents the amount of rain on the given day and month. For example,

    `rainfall[1][15]` is the amount of rain on Jan. 15
    `rainfall[12][25]` is the amount of rain on Dec. 25

    The array can be declared as follows:

```
double[][] rainfall = new double[13][32];
```

    This creates 13 rows indexed from `0` to `12` and 32 columns indexed from `0` to `31`, all initialized to `0.0`. Row `0` and column `0` will be ignored. Column `31` in row `4` will be ignored, since April 31 is not a valid day. In years that are not leap years, columns `29`, `30`, and `31` in row `2` will be ignored since Feb. 29, 30, and 31 are not valid days.

    Consider the method `averageRainfall` below:

```
/** Precondition:
 *   - rainfall is initialized with values representing amounts
 *     of rain on all valid days.
 *   - Invalid days are initialized to 0.0.
 *   - Feb 29 is not a valid day.
 *  Postcondition: Returns average rainfall for the year.
 */
public double averageRainfall(double rainfall[][])
{
    double total = 0.0;
    /* more code */
}
```

    Which of the following is a correct replacement for `/* more code */` so that the postcondition for the method is satisfied?

I
```
for (int month = 1; month < rainfall.length; month++)
    for (int day = 1; day < rainfall[month].length; day++)
        total += rainfall[month][day];
return total / (13 * 32);
```

II
```
for (int month = 1; month < rainfall.length; month++)
    for (int day = 1; day < rainfall[month].length; day++)
        total += rainfall[month][day];
return total / 365;
```

III
```
for (double[] month : rainfall)
    for (double rainAmt : month)
        total += rainAmt;
return total / 365;
```

(A) None
(B) I only
(C) II only
(D) III only
(E) II and III only

36. This question is based on the `Point` class below:

```
public class Point
{
    /** The coordinates. */
    private int x;
    private int y;

    public Point (int xValue, int yValue)
    {
        x = xValue;
        y = yValue;
    }

    /** @return the x-coordinate of this point */
    public int getx()
    { return x; }

    /** @return the y-coordinate of this point */
    public int gety()
    { return y; }

    /** Set x and y to new_x and new_y. */
    public void setPoint(int new_x, int new_y)
    {
        x = new_x;
        y = new_y;
    }

    //Other methods are not shown.
}
```

The method `changeNegs` below takes a matrix of `Point` objects as parameter and replaces every `Point` that has as least one negative coordinate with the `Point` (0,0).

```
/** @param pointMat the matrix of points
 *  Precondition: pointMat is initialized with Point objects.
 *  Postcondition: Every point with at least one negative coordinate
 *                 has been changed to have both coordinates
 *                 equal to zero.
 */
public static void changeNegs (Point [][] pointMat)
{
    /* code */
}
```

Which is a correct replacement for `/* code */`?

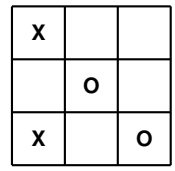
I
```
for (int r = 0; r < pointMat.length; r++)
    for (int c = 0; c < pointMat[r].length; c++)
        if (pointMat[r][c].getx() < 0
            || pointMat[r][c].gety() < 0)
               pointMat[r][c].setPoint(0, 0);
```

II
```
for (int c = 0; c < pointMat[0].length; c++)
    for (int r = 0; r < pointMat.length; r++)
        if (pointMat[r][c].getx() < 0
            || pointMat[r][c].gety() < 0)
               pointMat[r][c].setPoint(0, 0);
```

III
```
for (Point[] row : pointMat)
    for (Point p : row)
        if (p.getx() < 0 || p.gety() < 0)
               p.setPoint(0, 0);
```

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

37. A simple Tic-Tac-Toe board is a 3 × 3 array filled with either X's, O's, or blanks. Here is a class for a game of Tic-Tac-Toe:




```
public class TicTacToe
{
    private String[][] board;
    private static final int ROWS = 3;
    private static final int COLS = 3;

    /** Construct an empty board. */
    public TicTacToe()
    {
        board = new String[ROWS][COLS];
        for (int r = 0; r < ROWS; r++)
            for (int c = 0; c < COLS; c++)
                board[r][c] = " ";
    }

    /** @param r the row number
     *  @param c the column number
     *  @param symbol the symbol to be placed on board[r][c]
     *  Precondition: The square board[r][c] is empty.
     *  Postcondition: symbol placed in that square.
     */
    public void makeMove(int r, int c, String symbol)
    {
        board[r][c] = symbol;
    }

    /** Creates a string representation of the board, e.g.
     *    |o  |
     *    |xx |
     *    |  o|
     * @return the string representation of board
     */
    public String toString()
    {
        String s = "";     //empty string
        /* more code */
        return s;
    }
}
```

Which segment represents a correct replacement for `/* more code */` for the `toString` method?

(A)
```
for (int r = 0; r < ROWS; r++)
{
    for (int c = 0; c < COLS; c++)
    {
        s = s + "|";
        s = s + board[r][c];
        s = s + "|\n";
    }
}
```

(B)
```
for (int r = 0; r < ROWS; r++)
{
    s = s + "|";
    for (int c = 0; c < COLS; c++)
    {
        s = s + board[r][c];
        s = s + "|\n";
    }
}
```

(C)
```
for (int r = 0; r < ROWS; r++)
{
    s = s + "|";
    for (int c = 0; c < COLS; c++)
        s = s + board[r][c];
}
s = s + "|\n";
```

(D)
```
for (int r = 0; r < ROWS; r++)
    s = s + "|";
for (int c = 0; c < COLS; c++)
{
    s = s + board[r][c];
    s = s + "|\n";
}
```

(E)
```
for (int r = 0; r < ROWS; r++)
{
    s = s + "|";
    for (int c = 0; c < COLS; c++)
        s = s + board[r][c];
    s = s + "|\n";
}
```