1. Consider the following method. (Matching 3 points each)

public static int mystery(int[] arr)

{

int x = 0;

for (int k = 0; k < arr.length; k = k + 2)

x = x + arr[k];

return x;

}

Assume that the array nums has been declared and initialized as follows.

int[] nums = {3, 6, 1, 0, 1, 4, 2};

What value will be returned as a result of the call mystery(nums) ?

(A) 5

(B) 6

(C) 7

(D) 10

(E) 17

**Questions 2 refer to the following information.**

Consider the following partial class declaration.

public class SomeClass

{

private int myA;

private int myB;

private int myC;

// Constructor(s) not shown

public int getA()

{ return myA; }

public void setB(int value)

{ myB = value; }

}

2. The following declaration appears in another class.

SomeClass obj = new SomeClass();

Which of the following code segments will compile without error?

(A) int x = obj.getA();

(B) int x;

obj.getA(x);

(C) int x = obj.myA;

(D) int x = SomeClass.getA();

(E) int x = getA(obj);

4. Consider the following code segment.

int x = 7;

int y = 3;

if ((x < 10) && (y < 0))

System.out.println("Value is: " + (x \* y));

else

System.out.println("Value is: " + (x / y));

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

(A) Value is: 21

(B) Value is: 2.3333333

(C) Value is: 2

(D) Value is: 0

(E) Value is: 1

6. Consider the following method that is intended to determine if the double values d1 and d2 are close enough to be considered equal. For example, given a tolerance of 0.001, the values 54.32271 and 54.32294 would be considered equal.

/\*\* @return true if d1 and d2 are within the specified tolerance,

\* false otherwise

\*/

public boolean almostEqual(double d1, double d2, double tolerance)

{

**/\* *missing code* \*/**

}

Which of the following should replace /**\* *missing code* \*/** so that almostEqual will work as intended?

(A) return (d1 - d2) <= tolerance;

(B) return ((d1 + d2) / 2) <= tolerance;

(C) return (d1 - d2) >= tolerance;

(D) return ((d1 + d2) / 2) >= tolerance;

(E) return Math.abs(d1 - d2) <= tolerance;

8. Consider the following class declaration.

public class Student

{

private String myName;

private int myAge;

public Student()

{ /\* implementation not shown \*/ }

public Student(String name, int age)

{ /\* implementation not shown \*/ }

// No other constructors

}

Which of the following declarations will compile without error?

I. Student a = new Student();

II. Student b = new Student("Juan", 15);

III. Student c = new Student("Juan", "15");

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

9. Consider the following method that is intended to return the sum of the elements in the array key.

public static int sumArray(int[] key)

{

int sum = 0;

for (int i = 1; i <= key.length; i++)

{

**/\* *missing code* \*/**

}

return sum;

}

Which of the following statements should be used to replace /\* *missing code* \*/ so that sumArray will work as intended?

(A) sum = key[i];

(B) sum += key[i - 1];

(C) sum += key[i];

(D) sum += sum + key[i - 1];

(E) sum += sum + key[i];

12. Consider the following method.

public String mystery(String input)

{

String output = "";

for (int k = 1; k < input.length(); k = k + 2)

{

output += input.substring(k, k + 1);

}

return output;

}

What is returned as a result of the call mystery("computer") ?

(A) "computer"

(B) "cmue"

(C) "optr"

(D) "ompute"

(E) Nothing is returned because an IndexOutOfBoundsException is thrown.

13. Consider the following code segment.

int[] arr = {7, 2, 5, 3, 0, 10};

for (int k = 0; k < arr.length - 1; k++)

{

if (arr[k] > arr[k + 1])

System.out.print(k + " " + arr[k] + " ");

}

What will be printed as a result of executing the code segment?

(A) 0 2 2 3 3 0

(B) 0 7 2 5 3 3

(C) 0 7 2 5 5 10

(D) 1 7 3 5 4 3

(E) 7 2 5 3 3 0

17. Consider the following code segment.

int[] arr = {1, 2, 3, 4, 5, 6, 7};

for (int k = 3; k < arr.length - 1; k++)

arr[k] = arr[k + 1];

Which of the following represents the contents of arr as a result of executing the code segment?

(A) {1, 2, 3, 4, 5, 6, 7}

(B) {1, 2, 3, 5, 6, 7}

(C) {1, 2, 3, 5, 6, 7, 7}

(D) {1, 2, 3, 5, 6, 7, 8}

(E) {2, 3, 4, 5, 6, 7, 7}

19. Assume that a and b have been defined and initialized as int values. The expression

!(!(a != b ) && (b > 7))

is equivalent to which of the following?

(A) (a != b) || (b < 7)

(B) (a != b) || (b <= 7)

(C) (a == b) || (b <= 7)

(D) (a != b) && (b <= 7)

(E) (a == b) && (b > 7)

26. Assume that the array arr has been defined and initialized as follows.

int[] arr = /\* initial values for the array \*/ ;

Which of the following will correctly print all of the odd integers contained in arr but none of the even

integers contained in arr ?

(A) for (int x : arr)

if (x % 2 != 0)

System.out.println(x);

(B) for (int k = 1; k < arr.length; k++)

if (arr[k] % 2 != 0)

System.out.println(arr[k]);

(C) for (int x : arr)

if (x % 2 != 0)

System.out.println(arr[x]);

(D) for (int k = 0; k < arr.length; k++)

if (arr[k] % 2 != 0)

System.out.println(k);

(E) for (int x : arr)

if (arr[x] % 2 != 0)

System.out.println(arr[x]);

30. Consider the following method.

public static String scramble(String word, int howFar)

{

return word.substring(howFar + 1, word.length()) +

word.substring(0, howFar);

}

What value is returned as a result of the call scramble("compiler", 3)?

(A) "compiler"

(B) "pilercom"

(C) "ilercom"

(D) "ilercomp"

(E) No value is returned because an IndexOutOfBoundsException will be thrown.

33. Consider the following code segment.

int sum = 0;

int k = 1;

while (sum < 12 || k < 4)

sum += k;

System.out.println(sum);

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

(A) 6

(B) 10

(C) 12

(D) 15

(E) Nothing is printed due to an infinite loop.

35. Consider the following code segment.

int num = 2574;

int result = 0;

while (num > 0)

{

result = result \* 10 + num % 10;

num /= 10;

}

System.out.println(result);

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

(A) 2

(B) 4

(C) 18

(D) 2574

(E) 4752

Free Response Question. For the questions. **Complete any three of the following.**

1. ***Describe the task(s) you need to accomplish(1 pt)***
2. ***List the pre-conditions(1 pt)***
3. ***List the post-condition(1 pt)***
4. ***Start coding at the header(9 pts)***
5. Write a Boolean method isIncreasing() to determine if all of the values from the start to an ending index (stop) of the array **array** are increasing in value.

/\*\* @param **array** an array of positive integer values

\* @param **stop** the last index to check

\* **Precondition:** 0 <= stop < array.length

\* @return true if for each j such that 0 <= j < stop, array[j] < array[j + 1] ;

\* false otherwise

\*/

**public static boolean isIncreasing(int[] array, int stop)**

{ /\* implementation not shown \*/ }

1. Write a Boolean method isDecreasing() to determine if all of the values from a given starting index (start) to the end of the array are decreasing.

/\*\* @param array an array of positive integer values

\* @param start the first index to check

\* **Precondition:** 0 <= start < array.length - 1

\* @return true if for each j such that start <= j < array.length - 1,

\* array[j] > array[j + 1];

\* false otherwise

\*/

**public static boolean isDecreasing(int[] array, int start)**

{ /\* implementation not shown \*/ }

1. Write method getPeakIndex that returns the index of the first peak found in the parameter array, if one exists. A peak is defined as an element whose value is greater than the value of the element immediately before it and is also greater than the value of the element immediately after it. Method getPeakIndex starts at the beginning of the array and returns the index of the first peak that is found or -1 if no peak is found.

For example, the following table illustrates the results of several calls to getPeakIndex.

arr getPeakIndex(arr)

{11, 22, 33, 22, 11} 2

{11, 22, 11, 22, 11} 1

{11, 22, 33, 55, 77} -1

{99, 33, 55, 77, 120} -1

{99, 33, 55, 77, 55} 3

{33, 22, 11} -1

Complete method getPeakIndex below.

/\*\* @param array an array of positive integer values

\* **Precondition:** array.length > 0

\* @return the index of the first peak (local maximum) in the array, if it exists;

\* -1 otherwise

\*/

**public static int getPeakIndex(int[] array)**

{ /\* to be implemented in part (a) \*/ }

1. Write the method isMountain. Method isMountain returns true if the values in the parameter array are ordered as a mountain; otherwise, it returns false. The values in array are ordered as a mountain if all three of the following conditions hold.

• There must be a peak.

• The array elements with an index smaller than the peak’s index must appear in increasing order.

• The array elements with an index larger than the peak’s index must appear in decreasing order.

For example, the following table illustrates the results of several calls to isMountain.

arr isMountain(arr)

{1, 2, 3, 2, 1} true

{1, 2, 1, 2, 1} false

{1, 2, 3, 1, 5} false

{1, 4, 2, 1, 0} true

{9, 3, 5, 7, 5} false

{3, 2, 1} false

In writing isMountain, assume that getPeakIndex works as specified, regardless of what you wrote.

Complete method isMountain below.

/\*\* @param array an array of positive integer values

\* **Precondition:** array.length > 0

\* @return true if array contains values ordered as a mountain;

\* false otherwise

\*/

**public static boolean isMountain(int[] array)**