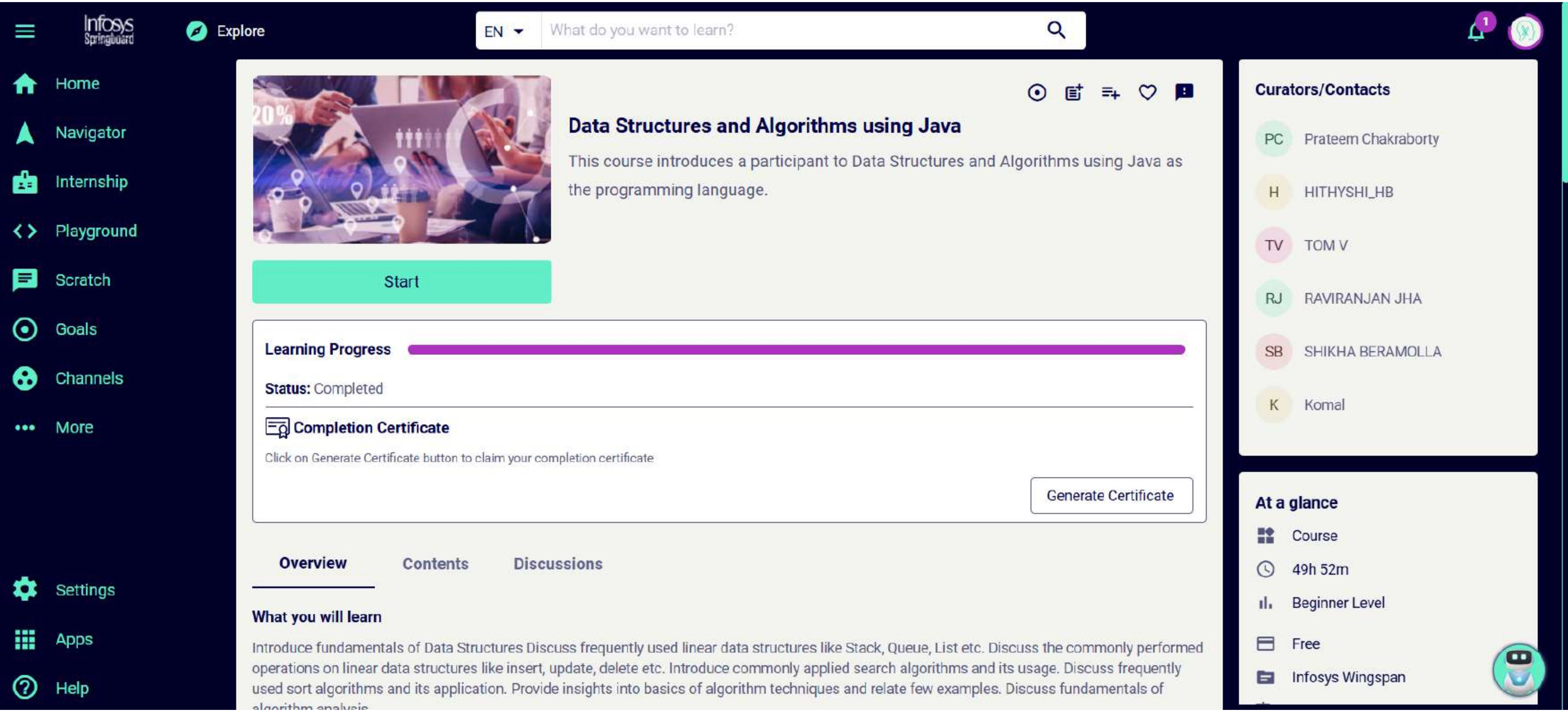


Data Structures and Algorithms using Java

This course introduces a participant to Data Structures and Algorithms using Java as the programming language.

Learning Progress

Status: Completed

Completion Certificate

Click on Generate Certificate button to claim your completion certificate

What you will learn

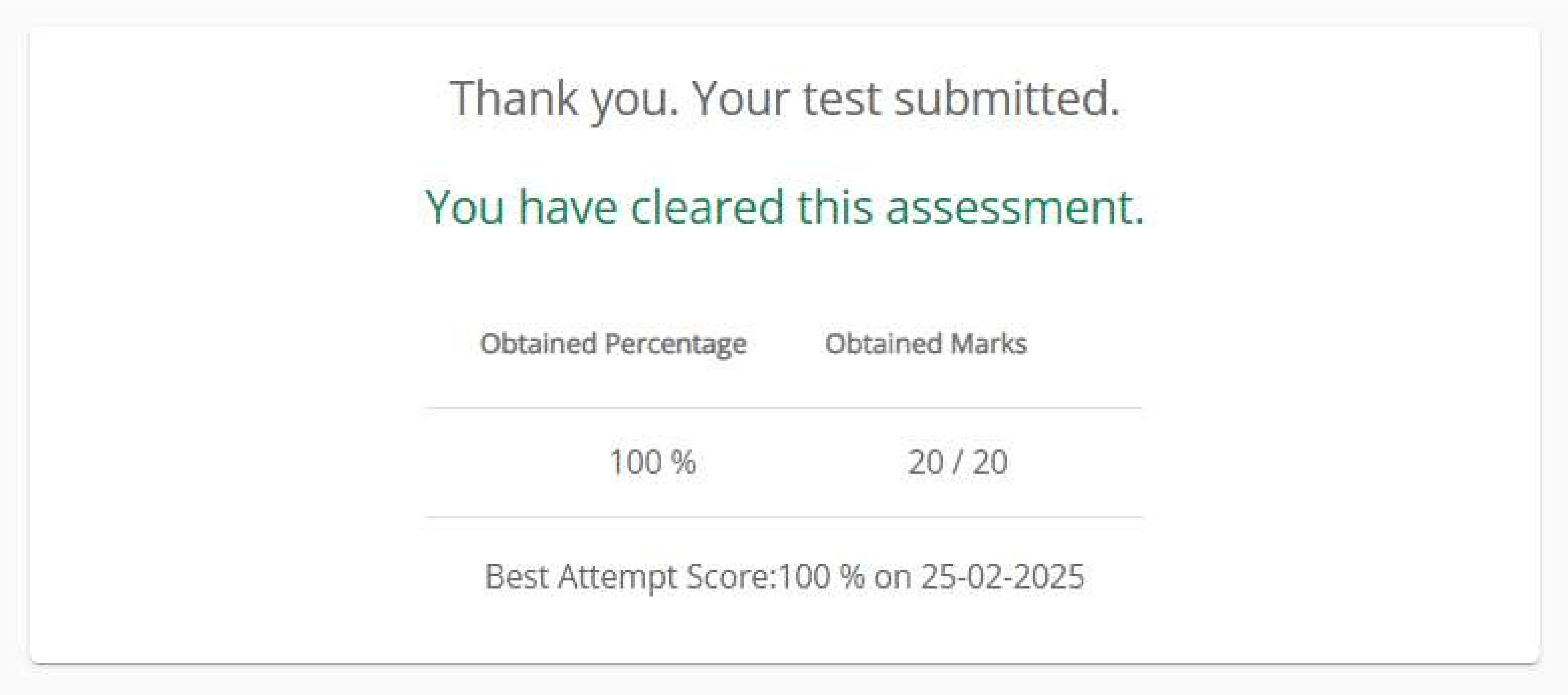
Introduce fundamentals of Data Structures Discuss frequently used linear data structures like Stack, Queue, List etc. Discuss the commonly performed operations on linear data structures like insert, update, delete etc. Introduce commonly applied search algorithms and its usage. Discuss frequently used sort algorithms and its application. Provide insights into basics of algorithm techniques and relate few examples. Discuss fundamentals of algorithm analysis

Curators/Contacts

PC Prateem Chakraborty
H HITHYSHI_HB
TV TOM V
RJ RAVIRANJAN JHA
SB SHIKHA BERAMOLLA
K Komal

At a glance

Course
49h 52m
Beginner Level
Free
Infosys Wingspan

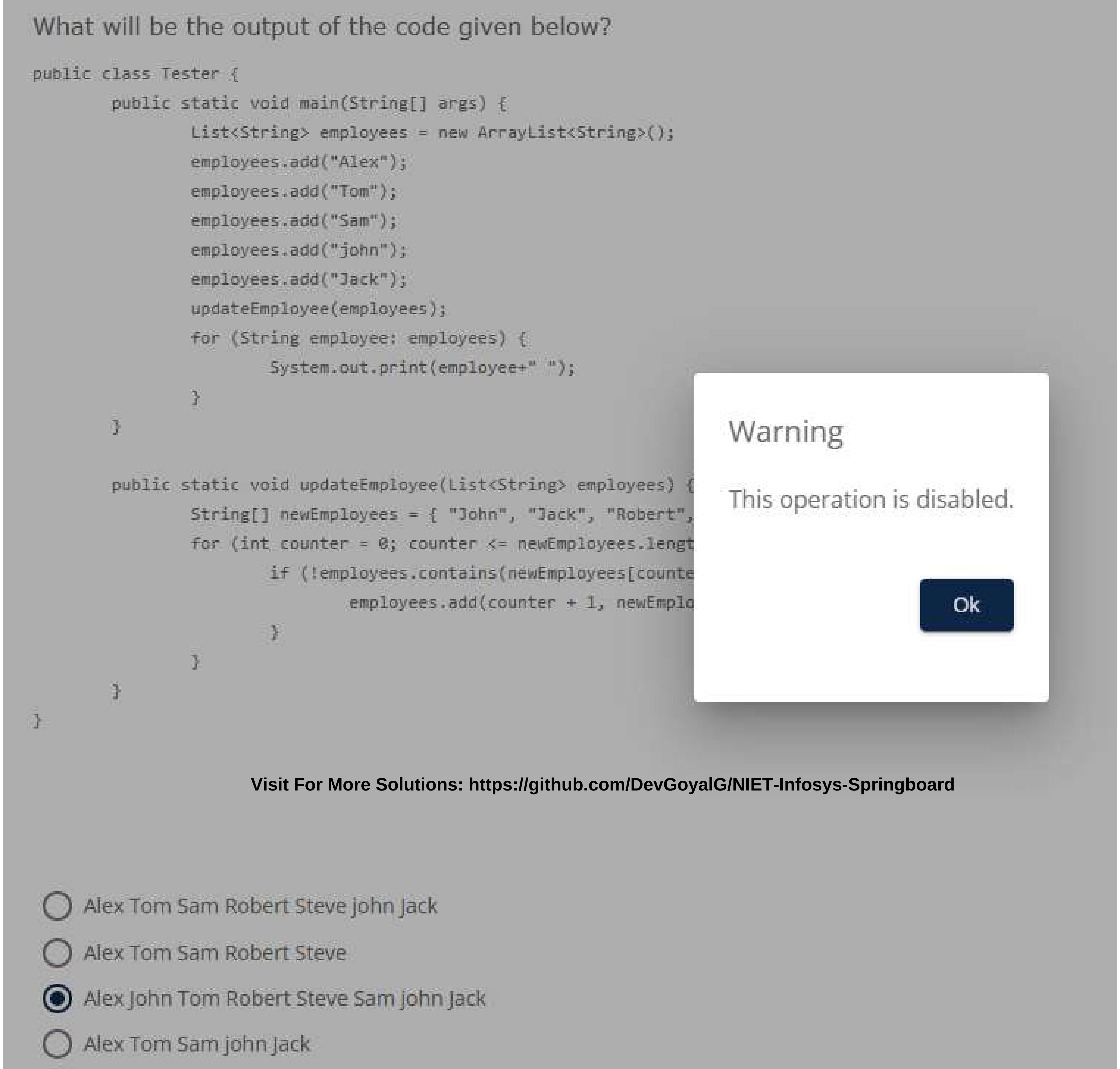


Thank you. Your test submitted.

You have cleared this assessment.

Obtained Percentage	Obtained Marks
100 %	20 / 20

Best Attempt Score:100 % on 25-02-2025



What will be the output of the code given below?

```
public class Tester {
        public static void main(String[] args) {
                List<String> employees = new ArrayList<String>();
                employees.add("Alex");
                employees.add("Tom");
                employees.add("Sam");
                employees.add("john");
                employees.add("Jack");
                updateEmployee(employees);
                for (String employee: employees) {
                        System.out.print(employee+" ");
                }
        }

        public static void updateEmployee(List<String> employees) {
                String[] newEmployees = { "John", "Jack", "Robert",
                for (int counter = 0; counter <= newEmployees.lengt
                        if (!employees.contains(newEmployees[counte
                                employees.add(counter + 1, newEmplo
                        }
                }
        }
}
```

Warning

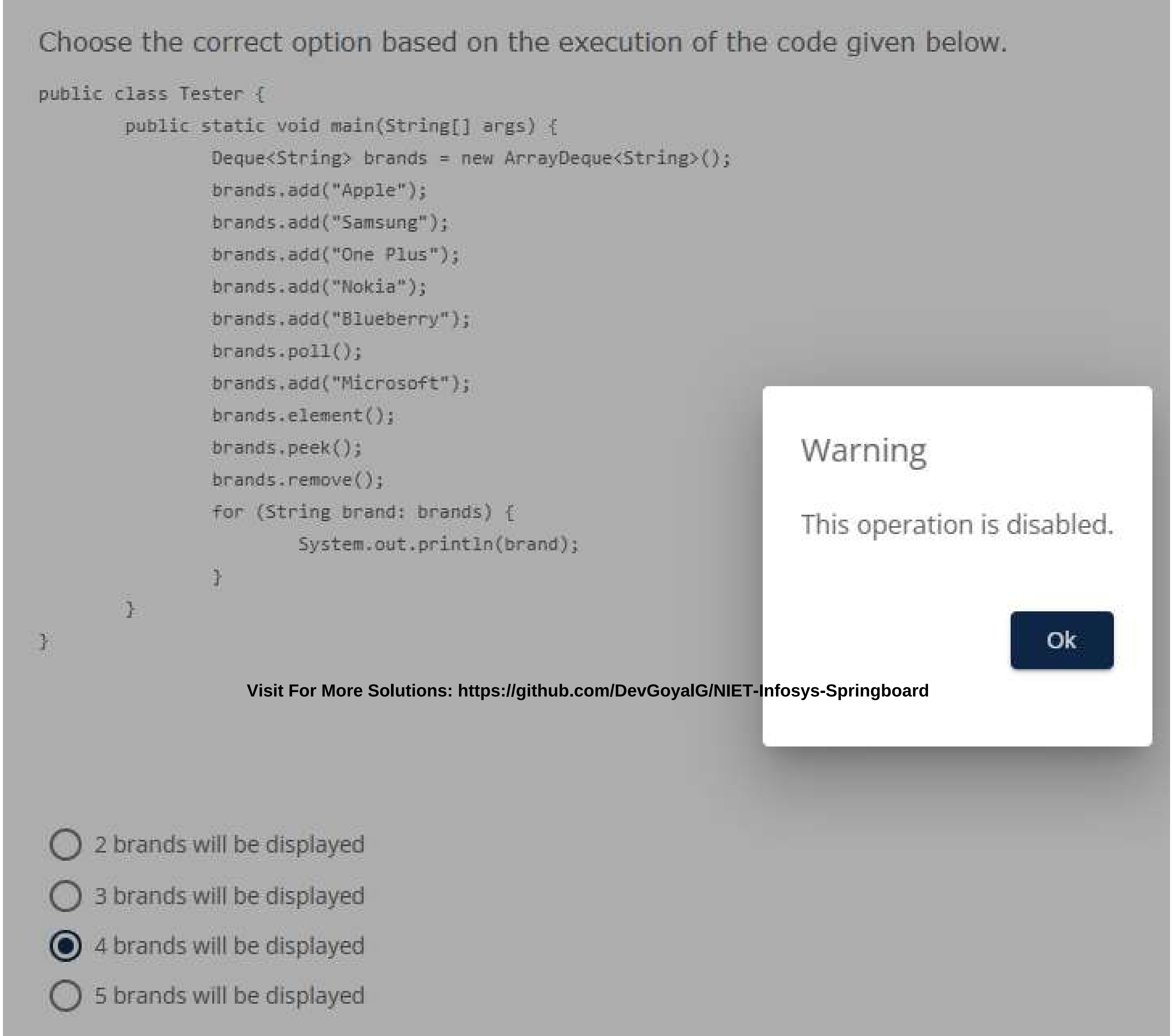
This operation is disabled.

○ Alex Tom Sam Robert Steve john Jack

○ Alex Tom Sam Robert Steve

● Alex John Tom Robert Steve Sam john Jack

○ Alex Tom Sam john Jack



Choose the correct option based on the execution of the code given below.

```
public class Tester {
        public static void main(String[] args) {
                Deque<String> brands = new ArrayDeque<String>();
                brands.add("Apple");
                brands.add("Samsung");
                brands.add("One Plus");
                brands.add("Nokia");
                brands.add("Blueberry");
                brands.poll();
                brands.add("Microsoft");
                brands.element();
                brands.peek();
                brands.remove();
                for (String brand: brands) {
                        System.out.println(brand);
                }
        }
}
```

Warning

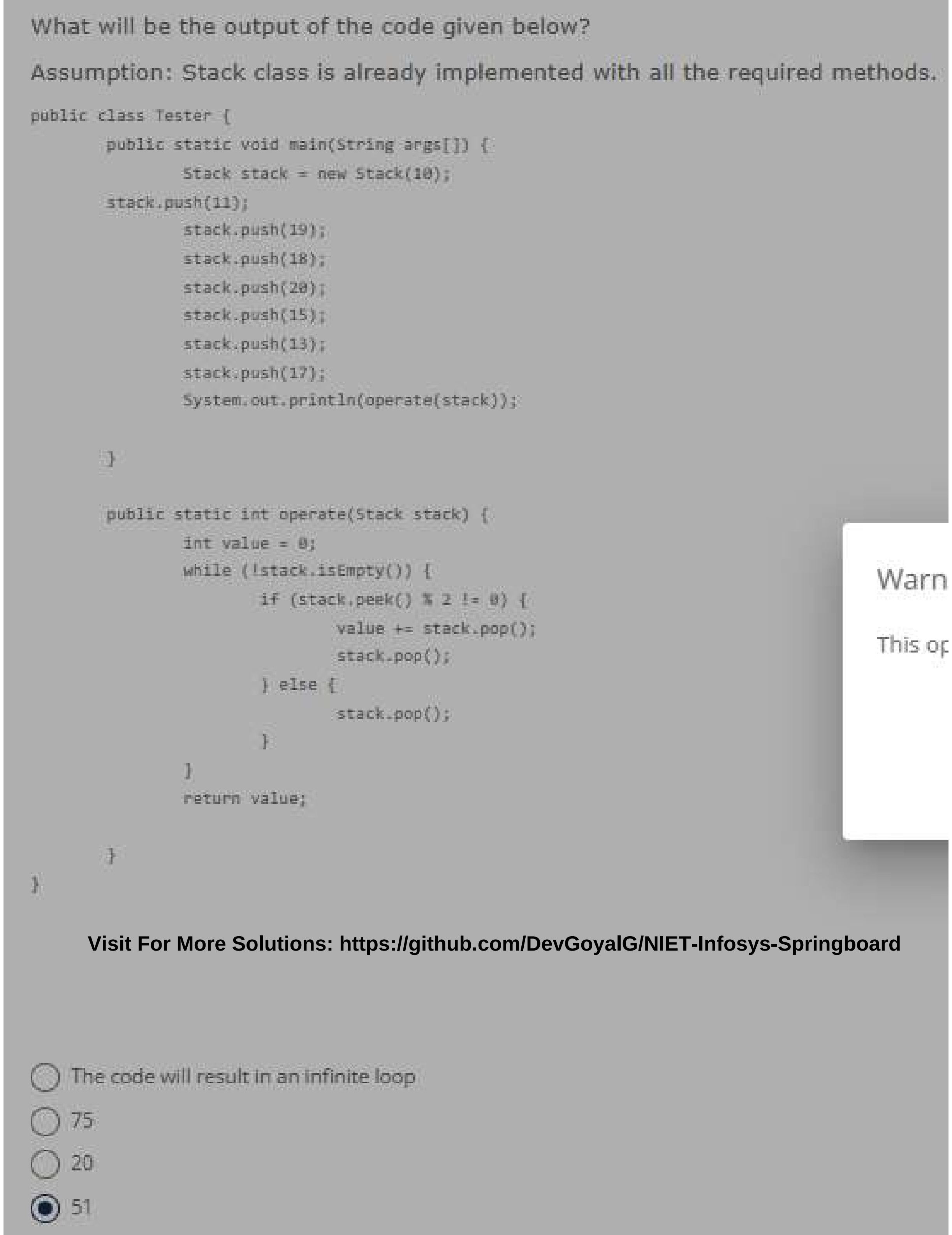
This operation is disabled.

○ 2 brands will be displayed

○ 3 brands will be displayed

● 4 brands will be displayed

○ 5 brands will be displayed



What will be the output of the code given below?

Assumption: Stack class is already implemented with all the required methods.

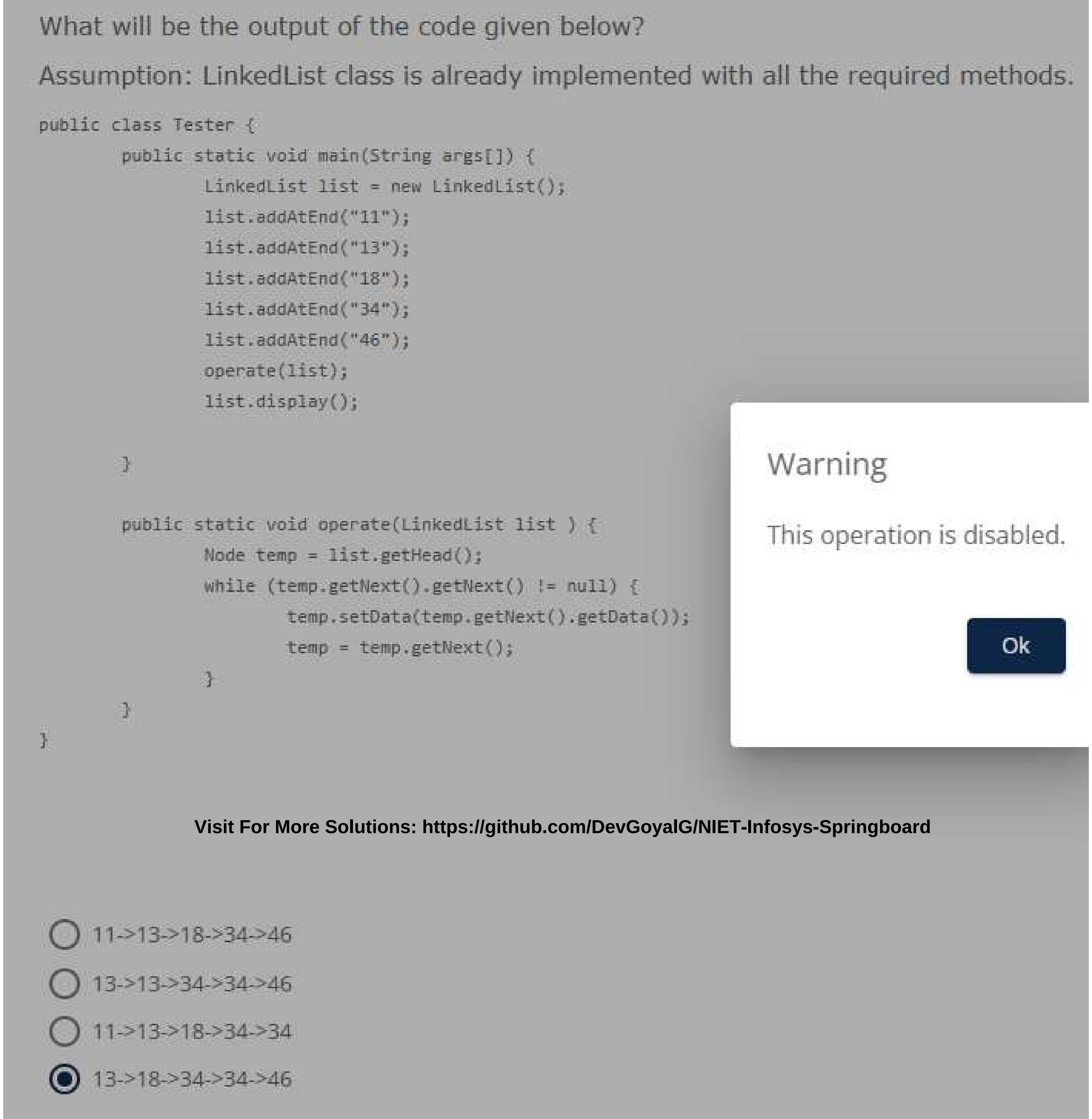
```
public class Tester {
        public static void main(String args[]) {
                Stack stack = new Stack(10);
        stack.push(11);
                stack.push(19);
                stack.push(18);
                stack.push(20);
                stack.push(15);
                stack.push(13);
                stack.push(17);
                System.out.println(operate(stack));

        }

        public static int operate(Stack stack) {
                int value = 0;
                while (!stack.isEmpty()) {
                        if (stack.peek() % 2 != 0) {
                                value += stack.pop();
                                stack.pop();
                        } else {
                                stack.pop();
                        }
                }
                return value;

        }
}
```

○ The code will result in an infinite loop

○ 75

○ 20

● 51



What will be the output of the code given below?

Assumption: LinkedList class is already implemented with all the required methods.

```
public class Tester {
        public static void main(String args[]) {
                LinkedList list = new LinkedList();
                list.addAtEnd("11");
                list.addAtEnd("13");
                list.addAtEnd("18");
                list.addAtEnd("34");
                list.addAtEnd("46");
                operate(list);
                list.display();

        }

        public static void operate(LinkedList list ) {
                Node temp = list.getHead();
                while (temp.getNext().getNext() != null) {
                        temp.setData(temp.getNext().getData());
                        temp = temp.getNext();
                }
        }
}
```

Warning

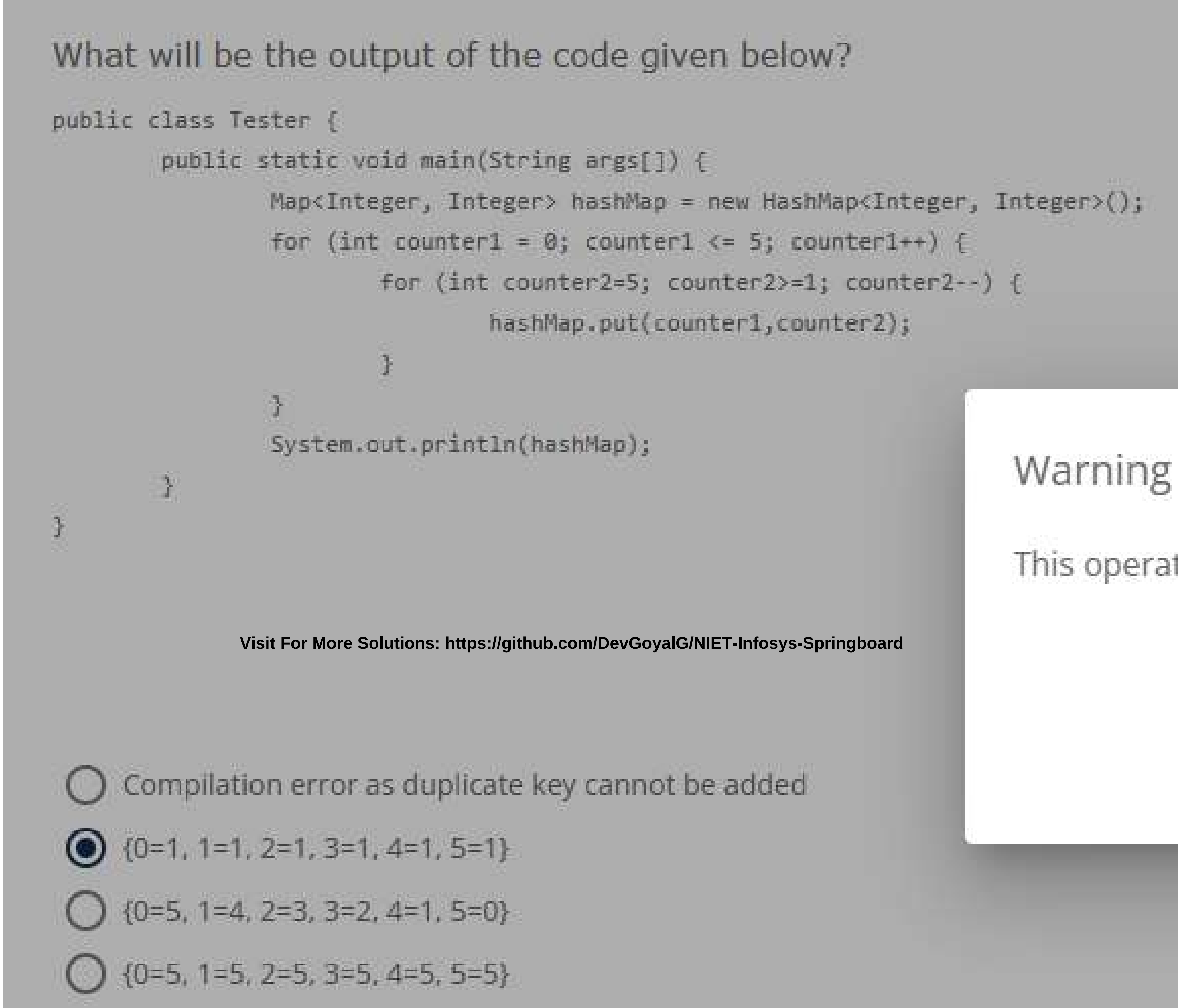
This operation is disabled.

○ 11->13->18->34->46

○ 13->13->34->34->46

○ 11->13->18->34->34

● 13->18->34->34->46



What will be the output of the code given below?

```
public class Tester {
    public static void main(String args[]) {
        Map<Integer, Integer> hashMap = new HashMap<Integer, Integer>();
        for (int counter1 = 0; counter1 <= 5; counter1++) {
            for (int counter2=5; counter2>=1; counter2--) {
                hashMap.put(counter1,counter2);
            }
        }
        System.out.println(hashMap);
    }
}
```

Warning

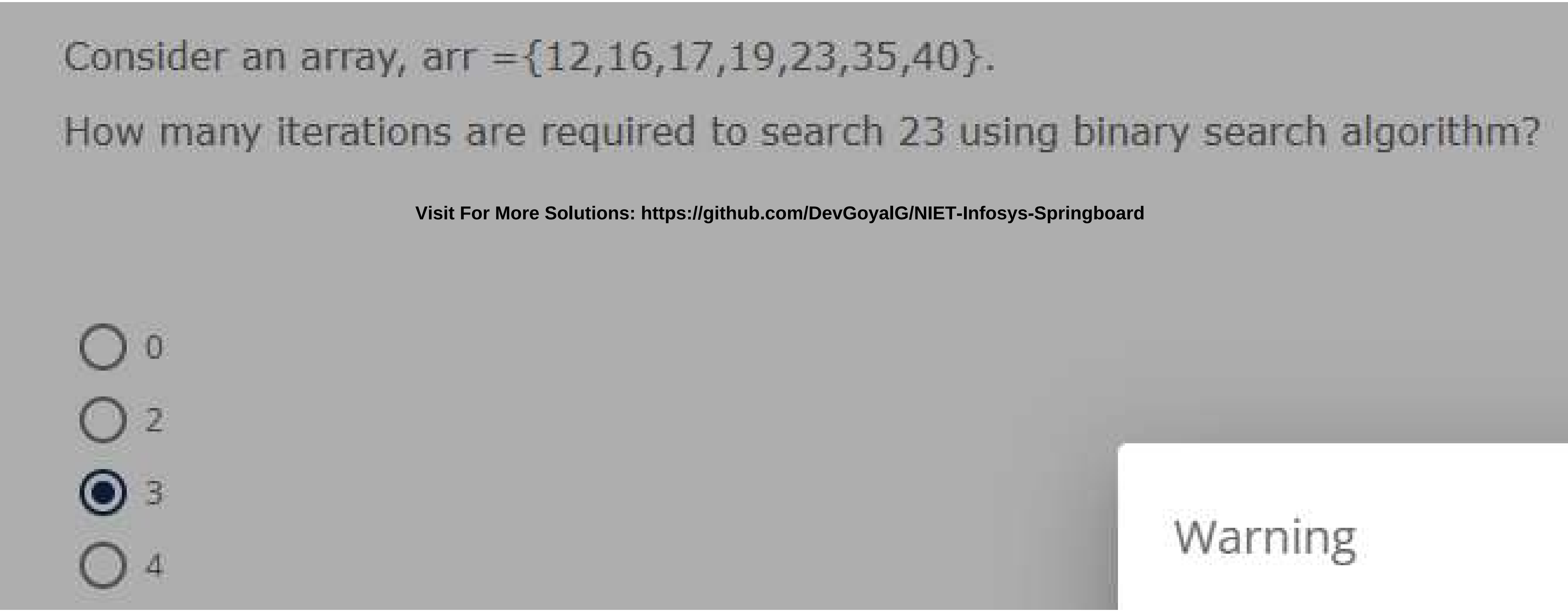
This operat

○ Compilation error as duplicate key cannot be added

● {0=1, 1=1, 2=1, 3=1, 4=1, 5=1}

○ {0=5, 1=4, 2=3, 3=2, 4=1, 5=0}

○ {0=5, 1=5, 2=5, 3=5, 4=5, 5=5}



Consider an array, arr ={12,16,17,19,23,35,40}.

How many iterations are required to search 23 using binary search algorithm?

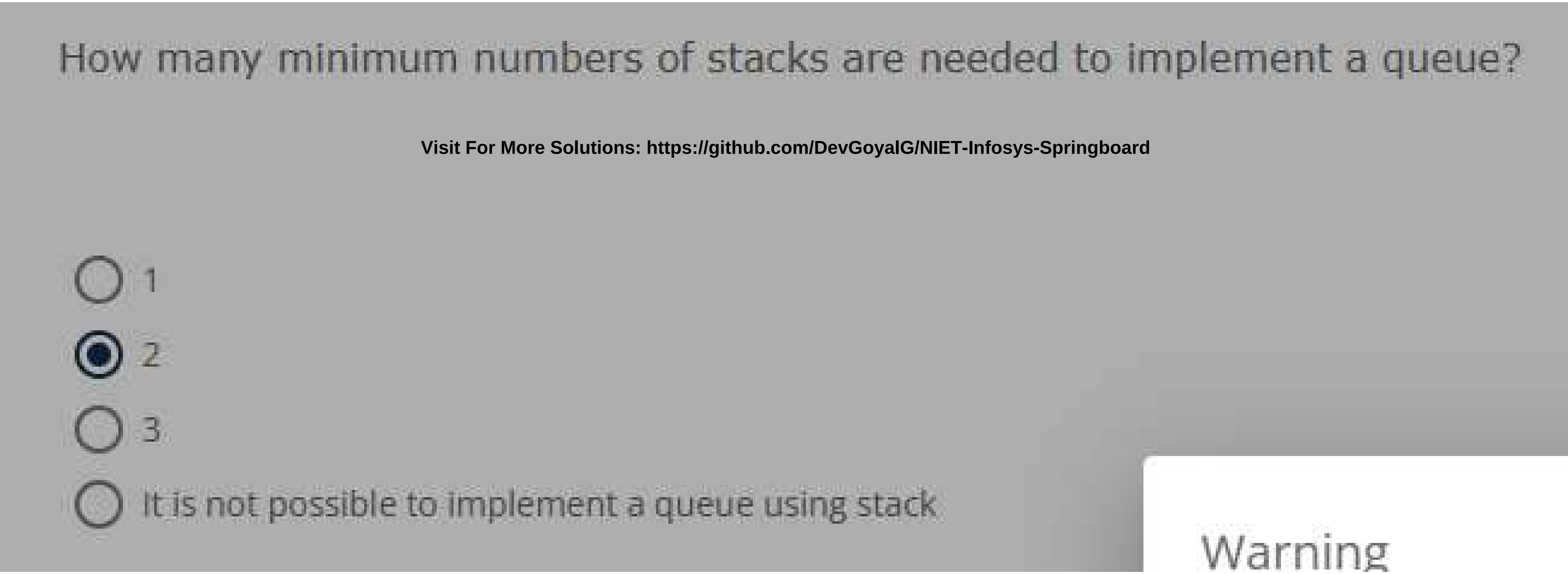
○ 0

○ 2

● 3

○ 4

Warning



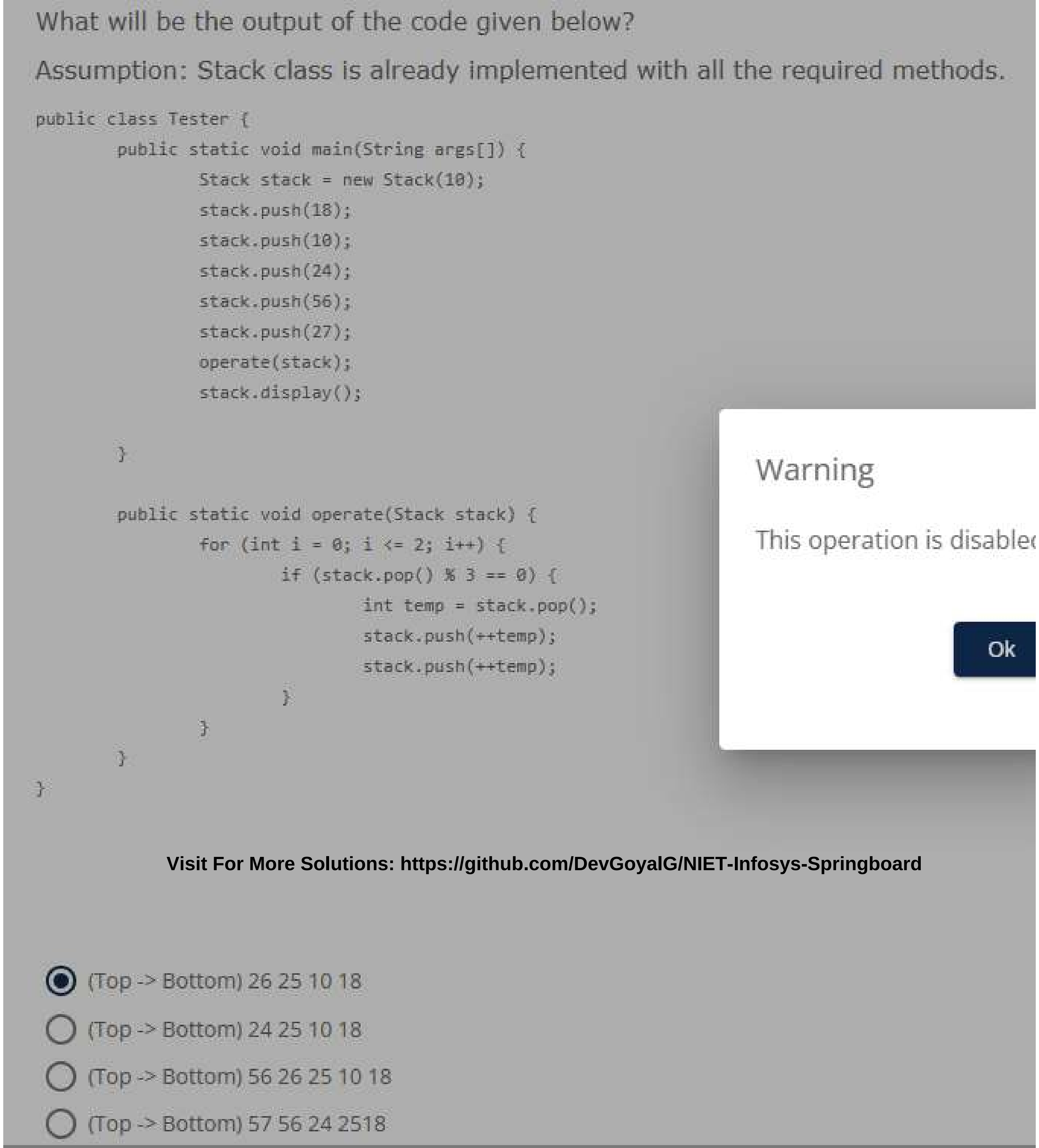
How many minimum numbers of stacks are needed to implement a queue?

○ 1

● 2

○ 3

○ It is not possible to implement a queue using stack



What will be the output of the code given below?

Assumption: Stack class is already implemented with all the required methods.

```
public class Tester {
        public static void main(String args[]) {
                Stack stack = new Stack(10);
                stack.push(18);
                stack.push(10);
                stack.push(24);
                stack.push(56);
                stack.push(27);
                operate(stack);
                stack.display();

        }

        public static void operate(Stack stack) {
                for (int i = 0; i <= 2; i++) {
                        if (stack.pop() % 3 == 0) {
                                int temp = stack.pop();
                                stack.push(++temp);
                                stack.push(++temp);
                        }
                }
        }
}
```

Warning

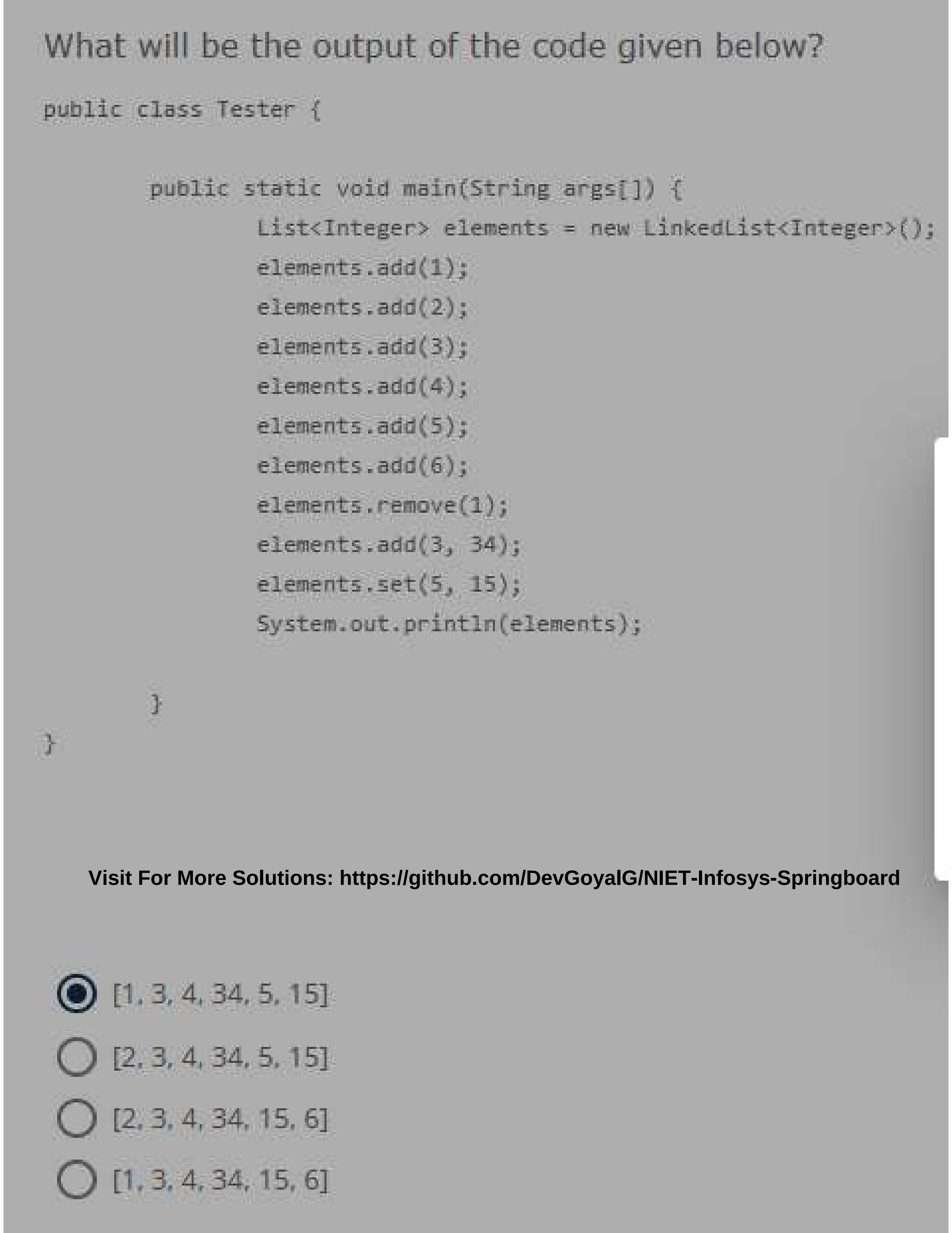
This operation is disable

● (Top -> Bottom) 26 25 10 18

○ (Top -> Bottom) 24 25 10 18

○ (Top -> Bottom) 56 26 25 10 18

○ (Top -> Bottom) 57 56 24 2518



What will be the output of the code given below?

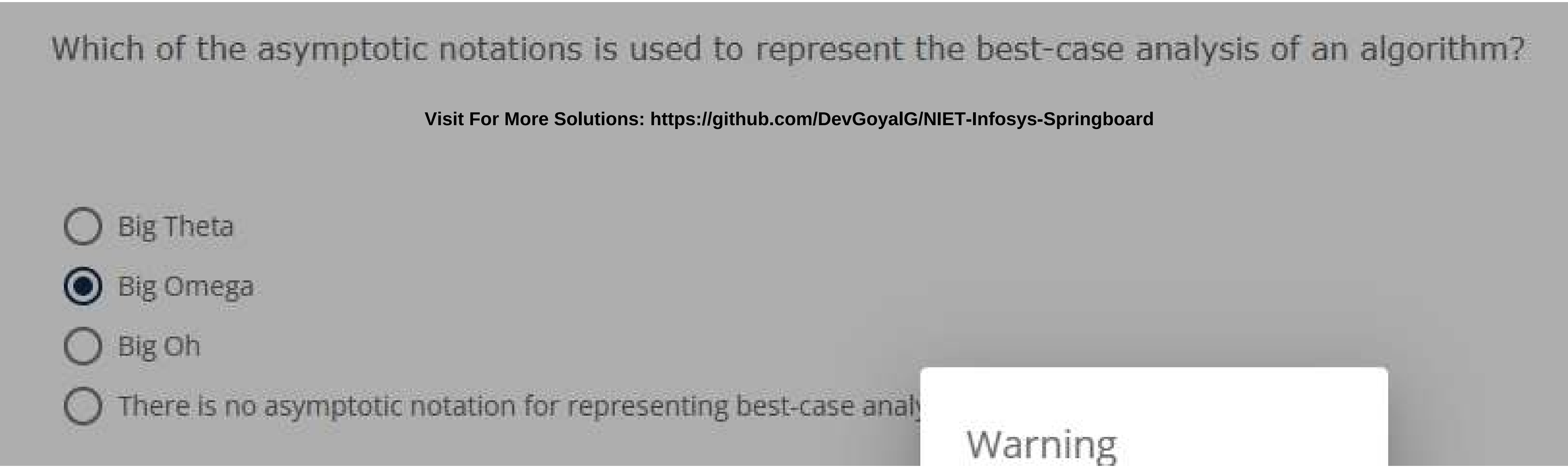
```
public class Tester {

        public static void main(String args[]) {
                List<Integer> elements = new LinkedList<Integer>();
                elements.add(1);
                elements.add(2);
                elements.add(3);
                elements.add(4);
                elements.add(5);
                elements.add(6);
                elements.remove(1);
                elements.add(3, 34);
                elements.set(5, 15);
                System.out.println(elements);

        }
}
```

● [1, 3, 4, 34, 5, 15]

○ [2, 3, 4, 34, 5, 15]

○ [2, 3, 4, 34, 15, 6]

○ [1, 3, 4, 34, 15, 6]



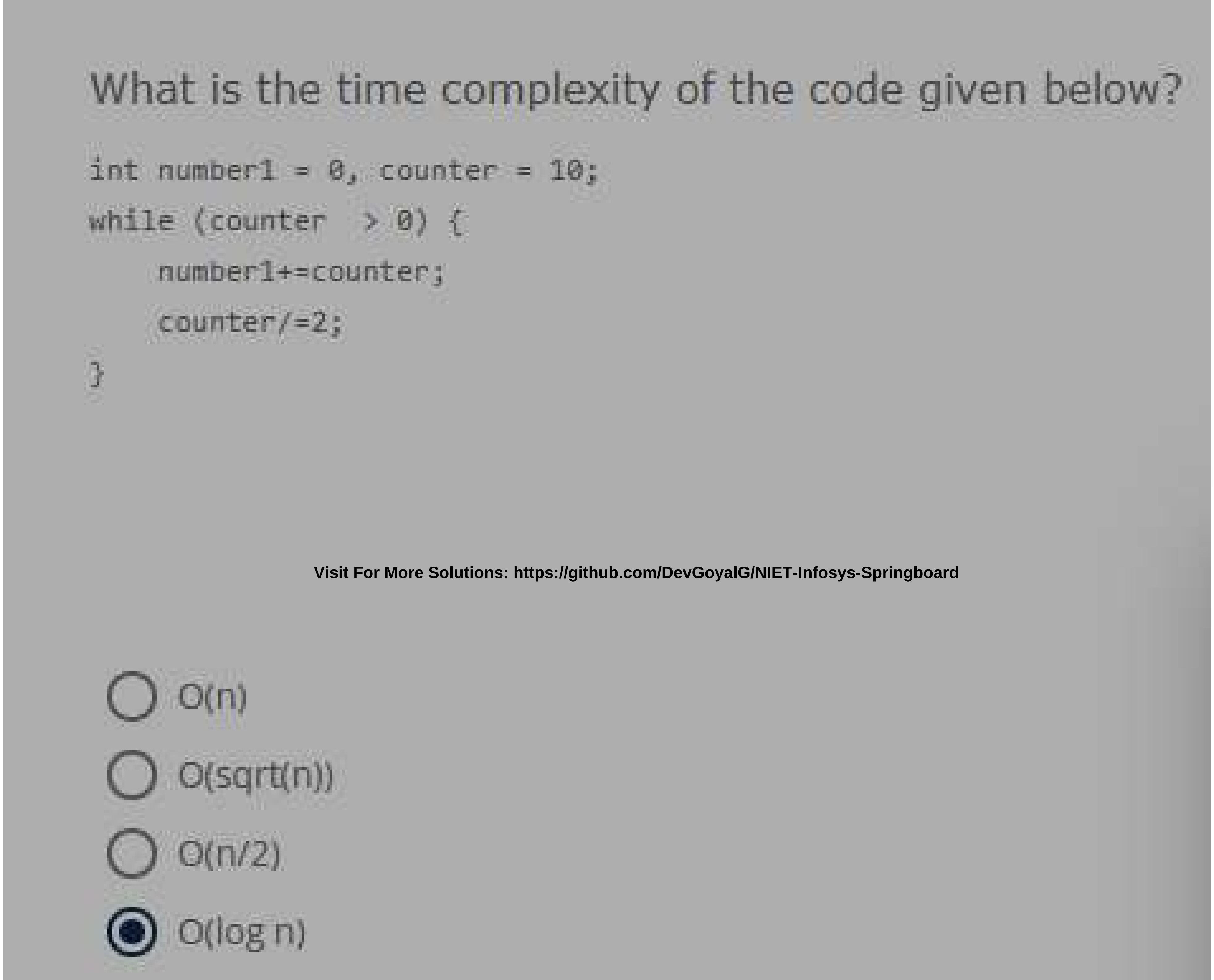
Which of the asymptotic notations is used to represent the best-case analysis of an algorithm?

○ Big Theta

● Big Omega

○ Big Oh

○ There is no asymptotic notation for representing best-case analy



What is the time complexity of the code given below?

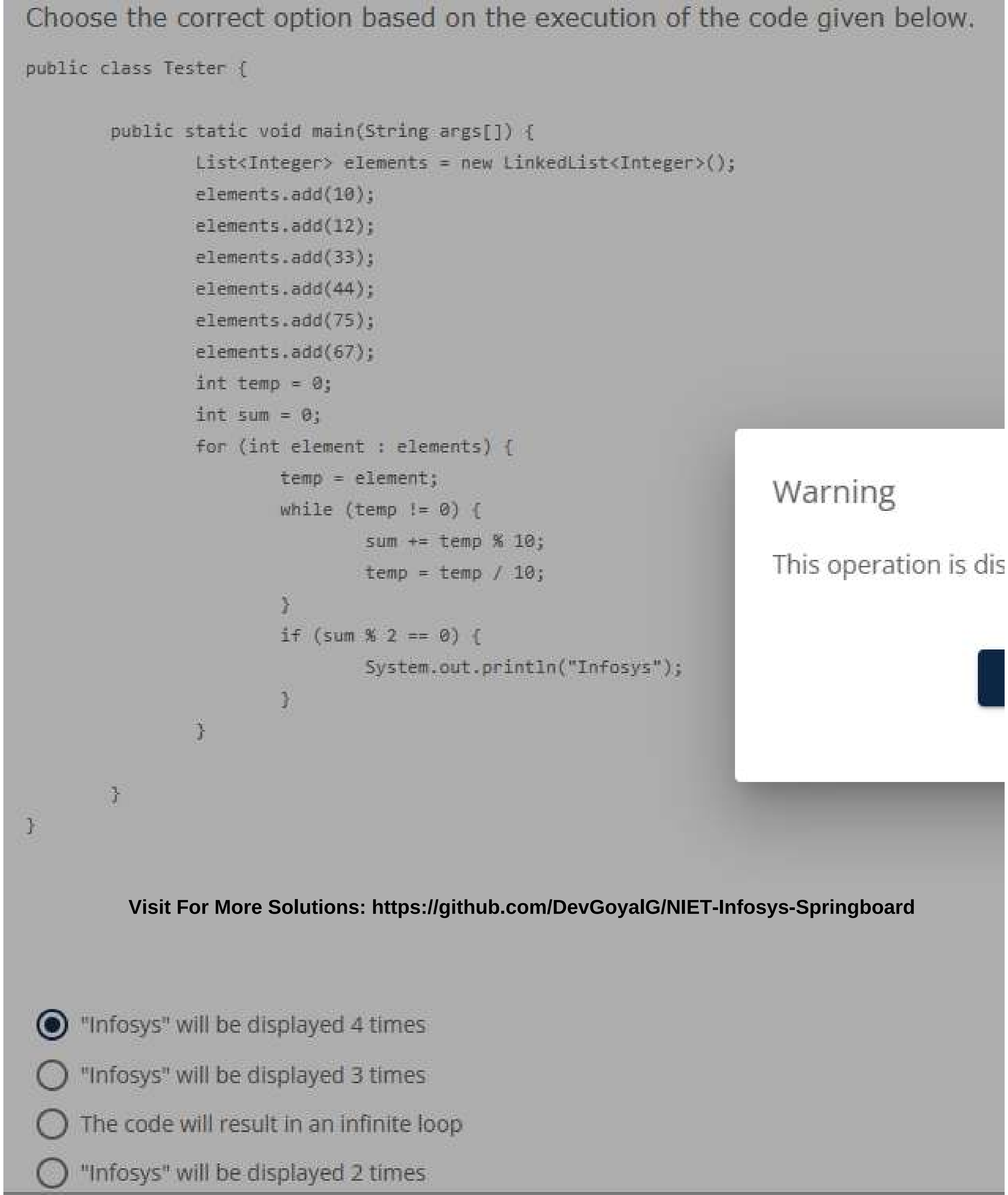
```
int number1 = 0, counter = 10;
while (counter  > 0) {
    number1+=counter;
    counter/=2;
}
```

○ O(n)

○ O(sqrt(n))

○ O(n/2)

● O(log n)



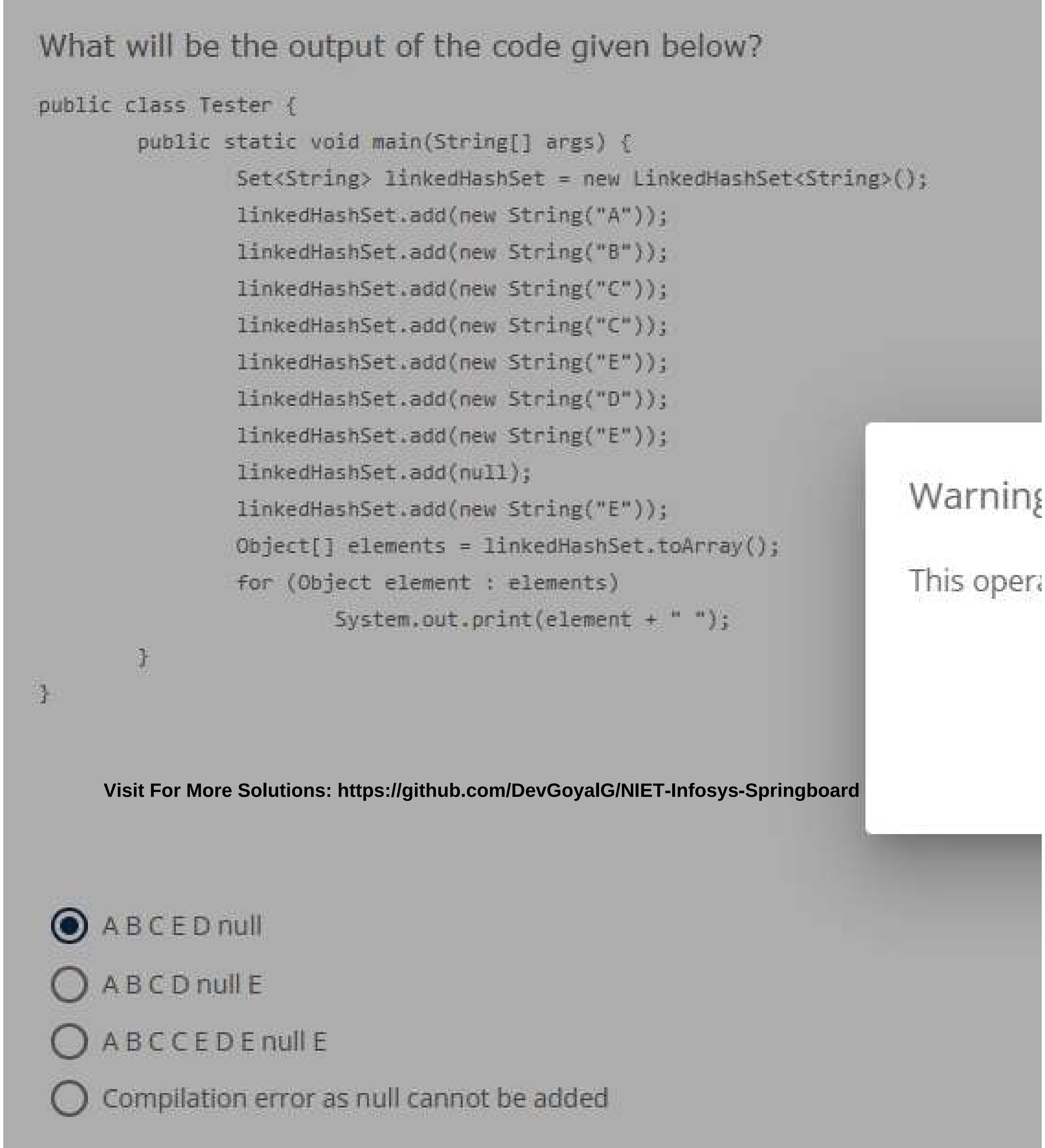
Choose the correct option based on the execution of the code given below.

```
public class Tester {

        public static void main(String args[]) {
                List<Integer> elements = new LinkedList<Integer>();
                elements.add(10);
                elements.add(12);
                elements.add(33);
                elements.add(44);
                elements.add(75);
                elements.add(67);
                int temp = 0;
                int sum = 0;
                for (int element : elements) {
                        temp = element;
                        while (temp != 0) {
                                sum += temp % 10;
                                temp = temp / 10;
                        }
                        if (sum % 2 == 0) {
                                System.out.println("Infosys");
                        }
                }

        }
}
```

Warning

This operation is dis

- ● "Infosys" will be displayed 4 times
- ○ "Infosys" will be displayed 3 times
- ○ The code will result in an infinite loop
- ○ "Infosys" will be displayed 2 times



What will be the output of the code given below?

```
public class Tester {
    public static void main(String[] args) {
        Set<String> linkedHashSet = new LinkedHashSet<String>();
        linkedHashSet.add(new String("A"));
        linkedHashSet.add(new String("B"));
        linkedHashSet.add(new String("C"));
        linkedHashSet.add(new String("C"));
        linkedHashSet.add(new String("E"));
        linkedHashSet.add(new String("D"));
        linkedHashSet.add(new String("E"));
        linkedHashSet.add(null);
        linkedHashSet.add(new String("E"));
        Object[] elements = linkedHashSet.toArray();
        for (Object element : elements)
            System.out.print(element + " ");
    }
}
```

Warning

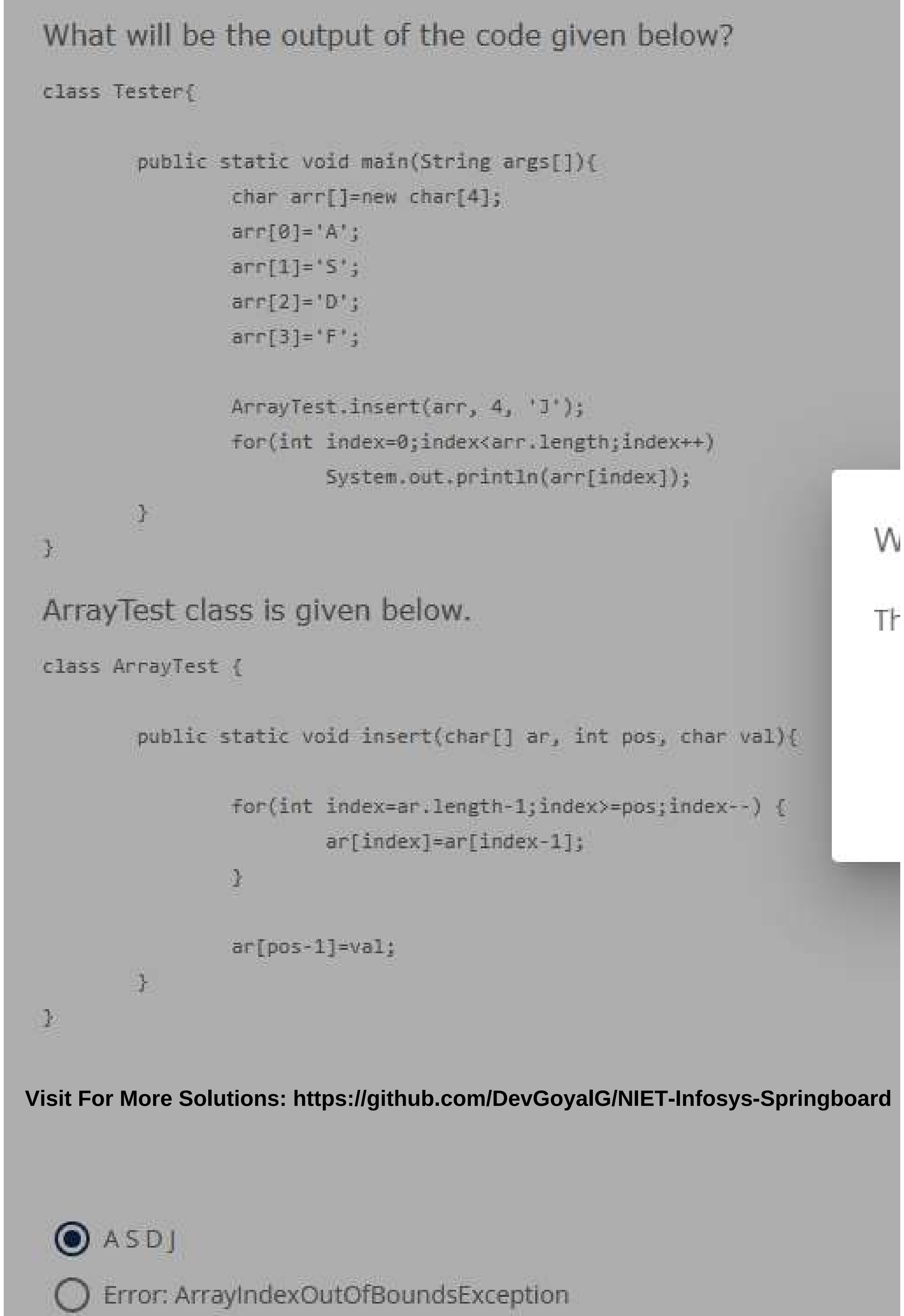
This oper

● A B C E D null

○ A B C D null E

○ A B C C E D E null E

○ Compilation error as null cannot be added



What will be the output of the code given below?

```
class Tester{

        public static void main(String args[]){
                char arr[]=new char[4];
                arr[0]='A';
                arr[1]='S';
                arr[2]='D';
                arr[3]='F';

                ArrayTest.insert(arr, 4, 'J');
                for(int index=0;index<arr.length;index++)
                        System.out.println(arr[index]);
        }
}
```

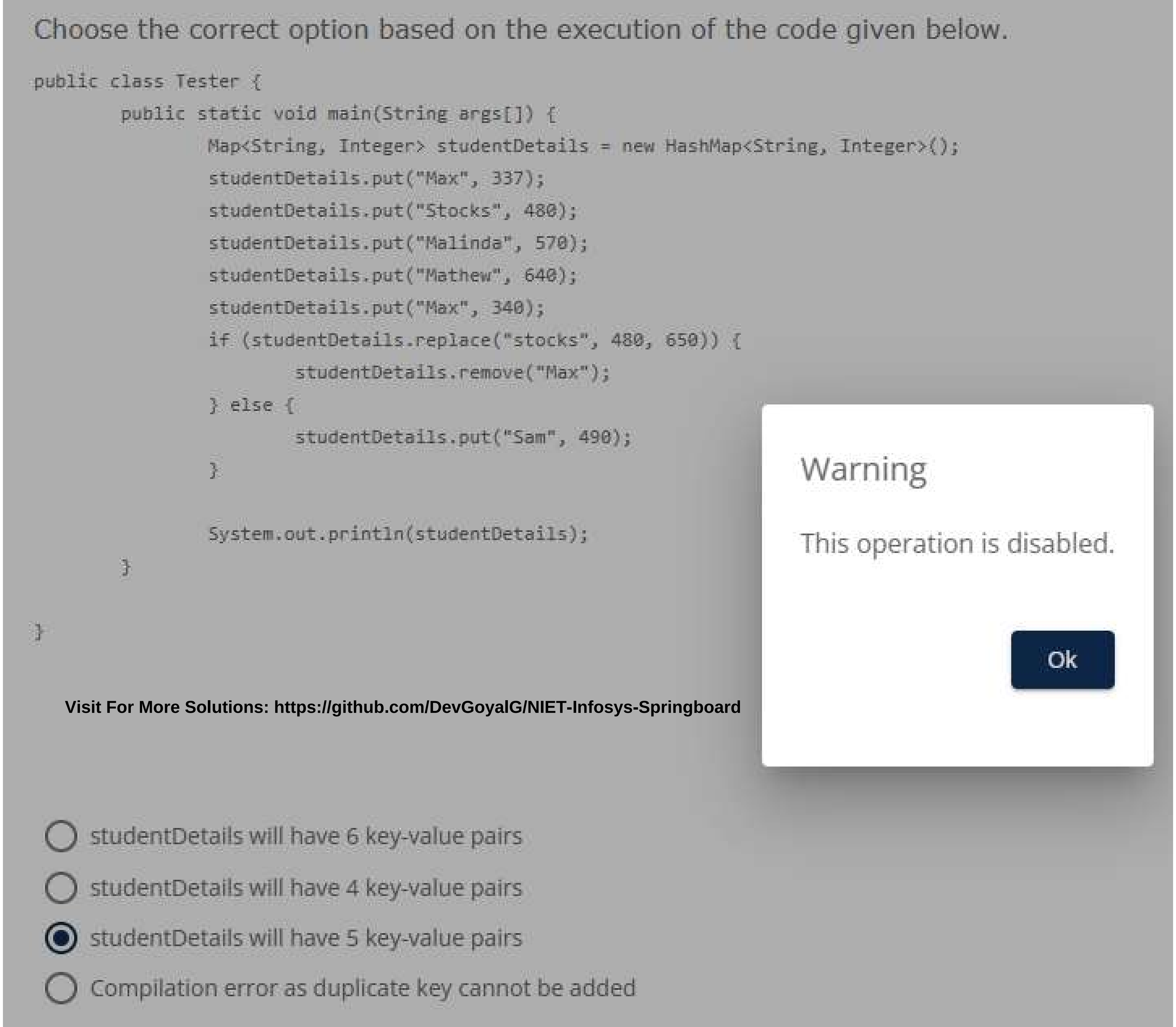
ArrayTest class is given below.

```
class ArrayTest {

        public static void insert(char[] ar, int pos, char val){

                for(int index=ar.length-1;index>=pos;index--) {
                        ar[index]=ar[index-1];
                }

                ar[pos-1]=val;
        }
}
```

Visit For More Solutions: https://github.com/DevGoyalG/NIET-Infosys-Springboard

● A S D J

○ Error: ArrayIndexOutOfBoundsException



Choose the correct option based on the execution of the code given below.

```
public class Tester {
        public static void main(String args[]) {
                Map<String, Integer> studentDetails = new HashMap<String, Integer>();
                studentDetails.put("Max", 337);
                studentDetails.put("Stocks", 480);
                studentDetails.put("Malinda", 570);
                studentDetails.put("Mathew", 640);
                studentDetails.put("Max", 340);
                if (studentDetails.replace("stocks", 480, 650)) {
                        studentDetails.remove("Max");
                } else {
                        studentDetails.put("Sam", 490);
                }

                System.out.println(studentDetails);
        }

}
```

Warning

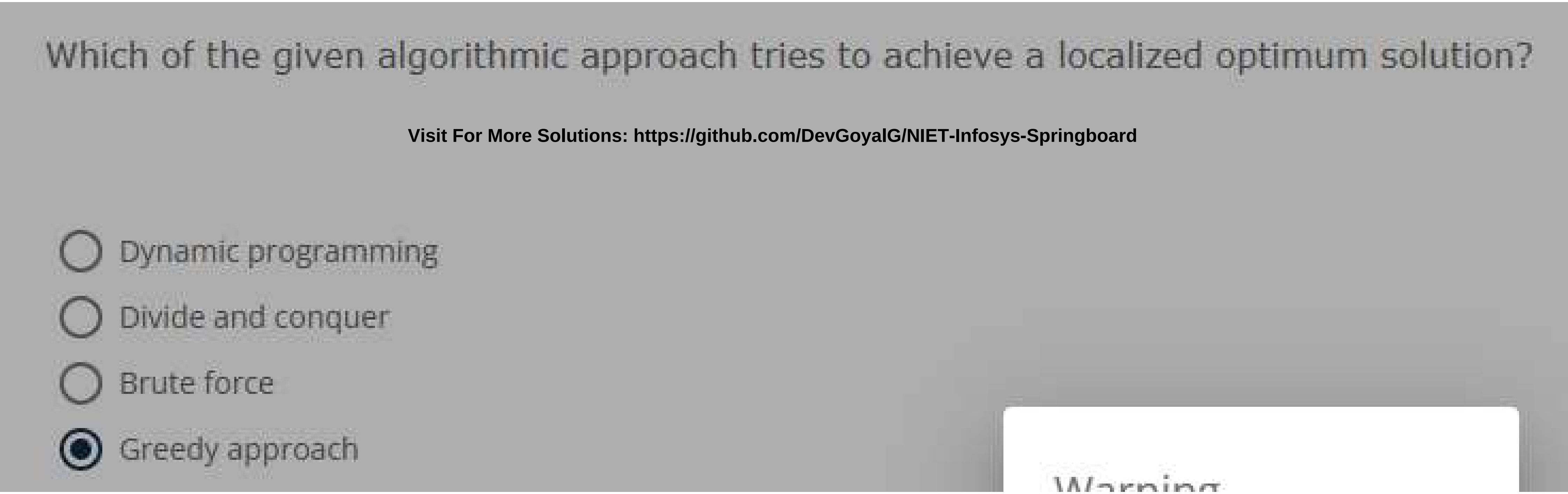
This operation is disabled.

○ studentDetails will have 6 key-value pairs

○ studentDetails will have 4 key-value pairs

● studentDetails will have 5 key-value pairs

○ Compilation error as duplicate key cannot be added



Which of the given algorithmic approach tries to achieve a localized optimum solution?

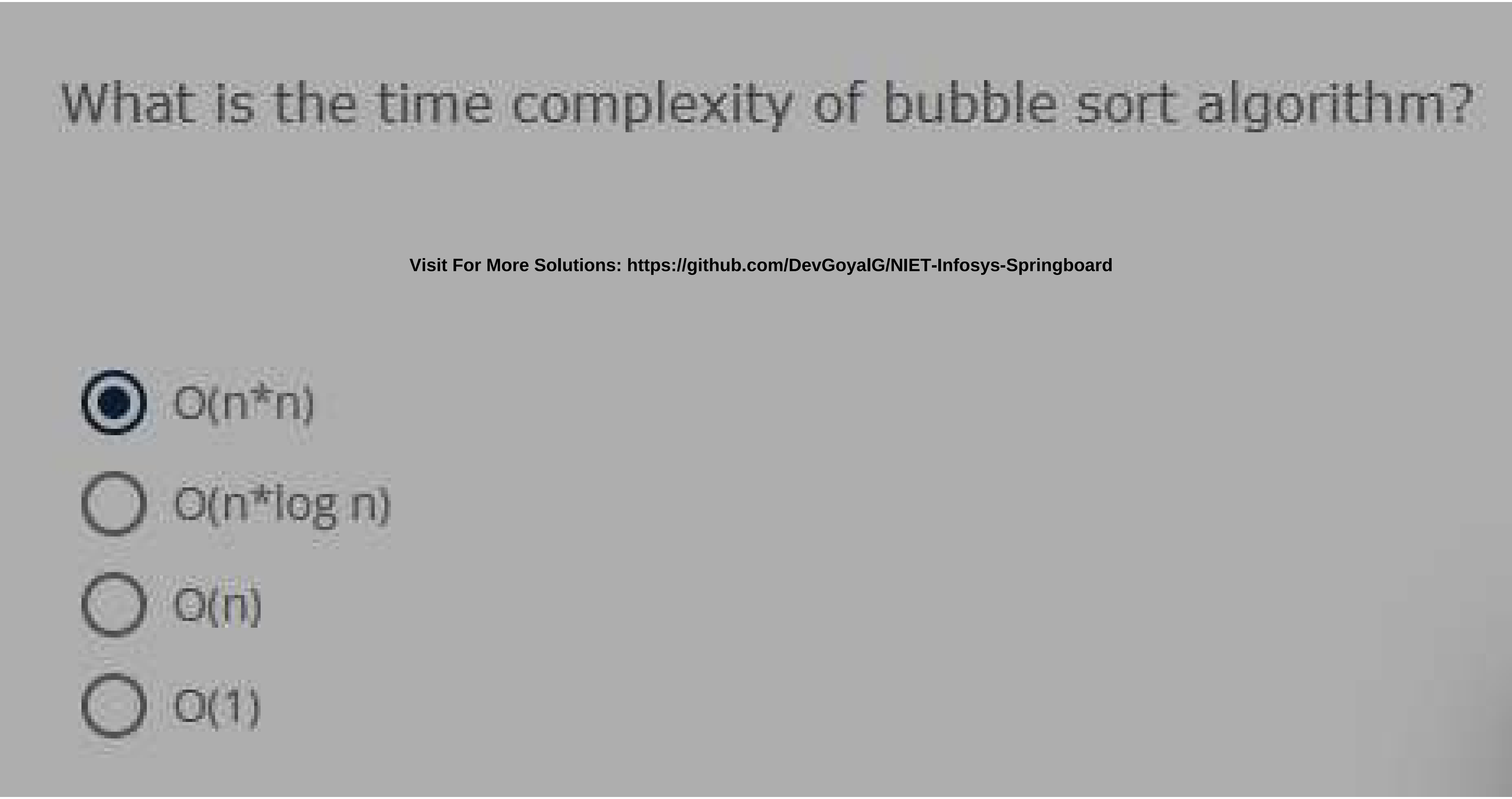
○ Dynamic programming

○ Divide and conquer

○ Brute force

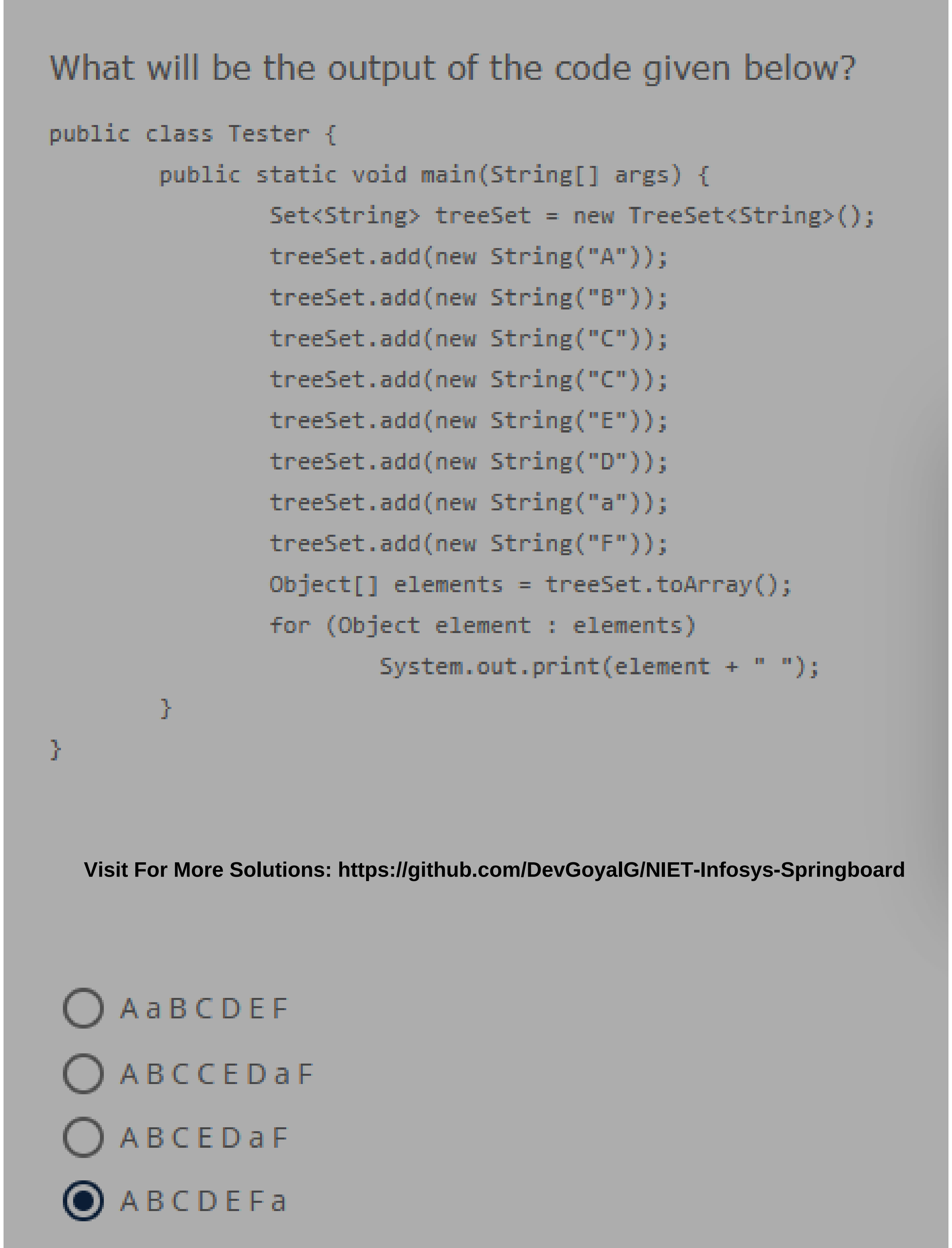
● Greedy approach

Warning



What is the time complexity of bubble sort algorithm?

- ● O(n*n)
- ○ O(n*log n)
- ○ O(n)
- ○ O(1)



What will be the output of the code given below?

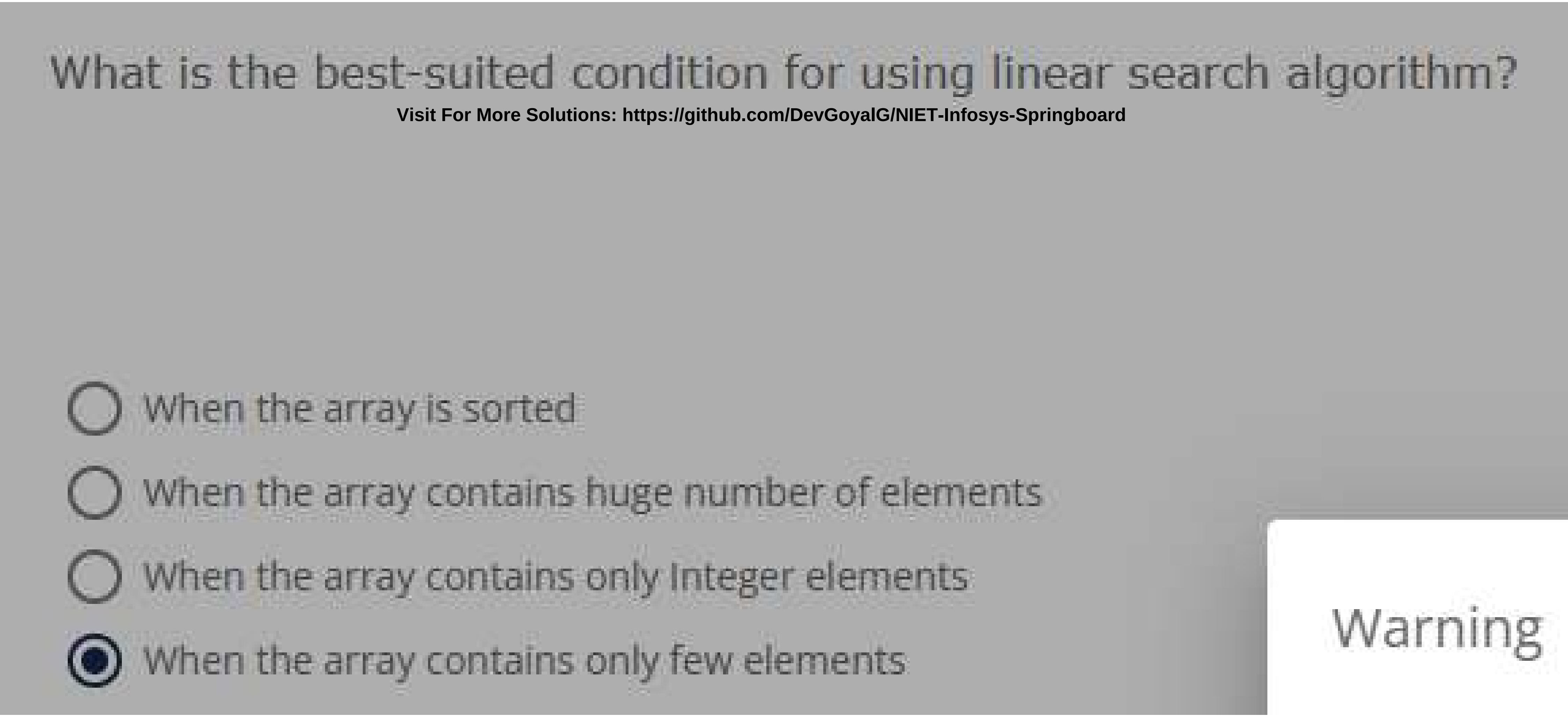
```
public class Tester {
        public static void main(String[] args) {
                Set<String> treeSet = new TreeSet<String>();
                treeSet.add(new String("A"));
                treeSet.add(new String("B"));
                treeSet.add(new String("C"));
                treeSet.add(new String("C"));
                treeSet.add(new String("E"));
                treeSet.add(new String("D"));
                treeSet.add(new String("a"));
                treeSet.add(new String("F"));
                Object[] elements = treeSet.toArray();
                for (Object element : elements)
                        System.out.print(element + " ");
        }
}
```

○ A a B C D E F

○ A B C C E D a F

○ A B C E D a F

● A B C D E F a



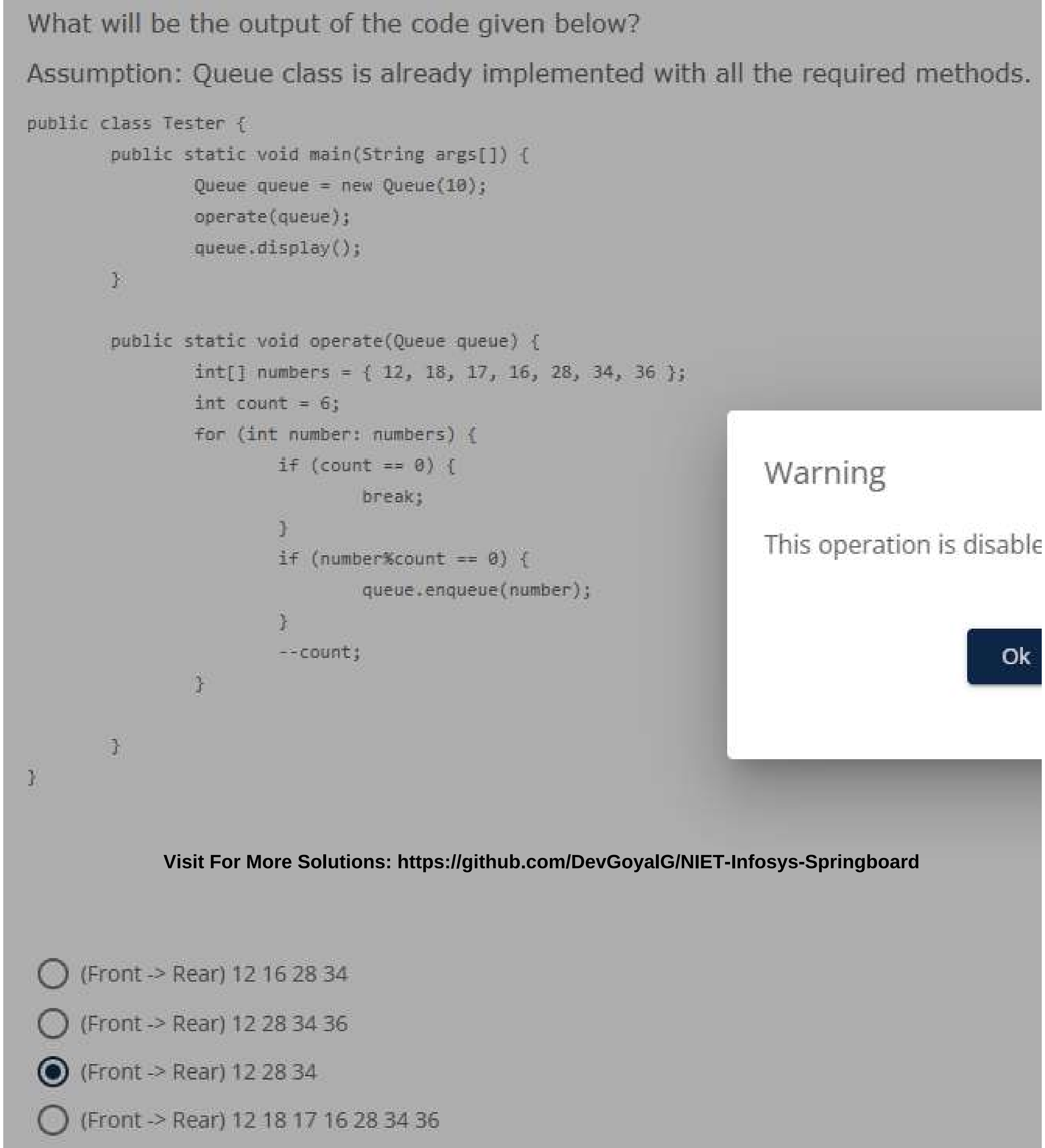
What is the best-suited condition for using linear search algorithm?

○ When the array is sorted

○ When the array contains huge number of elements

○ When the array contains only Integer elements

● When the array contains only few elements



What will be the output of the code given below?

Assumption: Queue class is already implemented with all the required methods.

```
public class Tester {
        public static void main(String args[]) {
                Queue queue = new Queue(10);
                operate(queue);
                queue.display();
        }

        public static void operate(Queue queue) {
                int[] numbers = { 12, 18, 17, 16, 28, 34, 36 };
                int count = 6;
                for (int number: numbers) {
                        if (count == 0) {
                                break;
                        }
                        if (number%count == 0) {
                                queue.enqueue(number);
                        }
                        --count;
                }

        }
}
```

Warning

This operation is disable

○ (Front -> Rear) 12 16 28 34

○ (Front -> Rear) 12 28 34 36

● (Front -> Rear) 12 28 34

○ (Front -> Rear) 12 18 17 16 28 34 36