COLLECTION - PRACTICE QUESTION

```
import java.util.*;


public class linkedList
{
	public static void main(String[] args)
	{
		List<String> list1 = new LinkedList<>();
		list1.add("Linked");
		list1.add("List");
		list1.add("Linked");
		list1.add("Map");
		list1.add("Collection");
		List<String> list2 = new LinkedList<>();
		list2.add("Linked");
		list1.removeAll(list2);
		for (String temp : list1)
			System.out.printf(temp + " ");
		System.out.println();
	}
}
```

Options:

1) Linked List Map Collection

2) List Map Collection

3) List Map Collection Linked

4) None of the above

2.

```
import java.util.*;
class QueueExample {
   public static void main(String args[]){
      PriorityQueue<String> queue=new PriorityQueue<String>();
      queue.add("Amit");
      queue.add("Rachit");
      queue.add("Rahul");
      System.out.println("head:"+queue.element());
      System.out.println("head:"+queue.peek());
      System.out.println("iterating the queue elements:");
      Iterator itr=queue.iterator();
      while(itr.hasNext()){
         System.out.println(itr.next());
      }
      queue.remove();
      queue.poll();
      System.out.println("after removing two elements:");
      Iterator<String> itr2=queue.iterator();
      while(itr2.hasNext()){
         System.out.println(itr2.next());
      }
   }
}
```

Options:

1) head:Amit
   head:Amit
   iterating the queue elements:
   Amit

Rachit

Rahul

after removing two elements:

Rachit

2) head:Amit

head:Amit

iterating the queue elements:

Amit

Rahul

Rachit

after removing two elements:

Rachit

3) head:Amit

head:Amit

iterating the queue elements:

Amit

Rachit

Rahul

after removing two elements:

Rahul

4) head:Amit

head:Amit

iterating the queue elements:

Amit

Rahul

Rachit

after removing two elements:

Rahul

3.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
public class MyListReplaceAll {
   public static void main(String a[]){
      List<String> list = new ArrayList<String>();
      list.add("java");
      list.add("unix");
      list.add("php");
      list.add("javascript");
      list.add("ruby");
      list.add(".net");
      list.add("java");
      System.out.println(list);
      Collections.replaceAll(list, "java", "ATM");
      System.out.println(list);
   }
}
```

Options:

1) [java, unix, php, javascript, ruby, .net, java]
   [ATM, unix, php, javascript, ruby, .net]
2) [java, unix, php, javascript, ruby, .net, java]
   [unix, php, javascript, ruby, .net, ATM]
3) [ATM, unix, php, javascript, ruby, .net, ATM]
   [java, unix, php, javascript, ruby, .net, java]
4) [java, unix, php, javascript, ruby, .net, java]
   [ATM, unix, php, javascript, ruby, .net, ATM]

4.

```
 import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class MyListFrequency {
   public static void main(String a[]){
      List<String> ll = new ArrayList<String>();
      ll.add("one");
      ll.add("two");
      ll.add("three");
      ll.add("four");
      ll.add("two");
      ll.add("three");
      ll.add("two");
      ll.add("one");
      System.out.println("Actual list: "+ll);
      System.out.println("Frequency of 'one': "+Collections.frequency(ll, "one"));
      System.out.println("Frequency of 'three': "+Collections.frequency(ll, "three"));
      System.out.println("Frequency of 'two': "+Collections.frequency(ll, "two"));
   }
}
```

Options:

1) Actual list: [one, two, three, four, two, three, two, one]
   Frequency of 'one': 2
   Frequency of 'three': 2
   Frequency of 'two': 3

2) Actual list: [one, two, three, four]

Frequency of 'one': 2

Frequency of 'three': 2

Frequency of 'two': 3

3) Actual list: [one, two, three, four]

Frequency of 'one': 1

Frequency of 'three': 1

Frequency of 'two': 1

4) Compilation Error

5.

```
import java.util.*;
public class Main {
  public static void main(String a[]){
    Vector<String> vct = new Vector<String>();
    vct.add("First");
    vct.add("Second");
    Enumeration<String> enm = vct.elements();
    while(enm.hasMoreElements()){
      System.out.println(enm.nextElement());
    }
    List<String> list = new ArrayList<String>();
    list.add("one");
    list.add("two");
    vct.addAll(list);
    System.out.println("After Copy: "+vct);
    Vector<String> copy = (Vector<String>) vct.clone();
    System.out.println("Cloned vector:"+copy);
  }
}
```

Options:

1) Compilation Error

2) First

  Second

  After Copy: [First, Second, one, two]

  Cloned vector:[First, one, Second, two]

3) First

  Second

  After Copy: [First, Second, one, two]

  Cloned vector:[First, Second, one, two]

4) Run Time Error

6.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

public class MyListDisjoint {

   public static void main(String a[]){

      List<String> sl = new ArrayList<String>();
      sl.add("java");
      sl.add("c++");
      sl.add("unix");
      List<String> tl = new ArrayList<String>();
      tl.add("job");
      tl.add("oracle");
```

```
        boolean isCommon = Collections.disjoint(sl,tl);
        System.out.println("Does not found any common elements? "+isCommon);
        tl.add("java");
        isCommon = Collections.disjoint(sl,tl);
        System.out.println("Does not found any common elements? "+isCommon);
    }
}
```

Options:

1) Does not found any common elements? true
   Does not found any common elements? false
2) Does not found any common elements? true
   Does not found any common elements? true
3) Does not found any common elements? false
   Does not found any common elements? true
4) RunTime Error

ANSWERS:

1. 2) List Map Collection
2. 3) head:Amit

   head:Amit

   iterating the queue elements:

   Amit

   Rachit

   Rahul

   after removing two elements:

   Rahul

3. 4) [java, unix, php, javascript, ruby, .net, java]
   [ATM, unix, php, javascript, ruby, .net, ATM]
4. 1) Actual list: [one, two, three, four, two, three, two, one]

   Frequency of 'one': 2

   Frequency of 'three': 2

   Frequency of 'two': 3

5. 3) First

   Second

   After Copy: [First, Second, one, two]

   Cloned vector:[First, Second, one, two]

6. 1) Does not found any common elements? true

   Does not found any common elements? False