Work Time: 25min

Please copy the subjects and then close your laptops.

Default (1p).

1 (3p). Given the following Java collection:

```
List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14,15);
```

Using Java functional style (Java streams), please write a Java stream program that is doing the following

a) Eliminate all the numbers which are multiple of 3 or multiple of 7.

b) Transform each remaining number into its predecessor multiplied by 11 (e.g., 4 is transformed into 33).

c) Compute the sum modulo 5 of the remaining numbers.

Solution:

Step 1: Remove all numbers that are multiples of 3 or 7.

- The numbers divisible by 3 are: {3, 6, 9, 12, 15}

- The numbers divisible by 7 are: {7, 14}

- The numbers remaining after removal: {1, 2, 4, 5, 8, 10, 11}

Step 2: Transform each remaining number into its predecessor multiplied by 11.

- (1 - 1) * 11 = 0

- (2 - 1) * 11 = 11

- (4 - 1) * 11 = 33

- (5 - 1) * 11 = 44

- (8 - 1) * 11 = 77

- (10 - 1) * 11 = 99

- (11 - 1) * 11 = 110

Resulting list: {0, 11, 33, 44, 77, 99, 110}

Step 3: Compute the sum modulo 5.

- Sum = 0 + 11 + 33 + 44 + 77 + 99 + 110 = 374

- 374 % 5 = 4

Final result: 4

2 (3p). Given the following four classes in Java:

```
class A implements D{...}   class B extends A implements D {...}

class C extends B implements D {...}   interface D {...}

class Amain{

       ...  method1(ArrayList<.......> list) {  if list.isEmpty() return null; else return list.get(0);}

       void method2(ArrayList<........>  list, C elem) {  list.add(elem);}

      void method3(C elem){

            ArrayList<A> listA=new ArrayList<A>(); listA.add(new B());listA.add(new C());

            ArrayList<B> listB = new ArrayList<B>(); listB.add(new B());listB.add(new C());

            ArrayList<C> listC = new ArrayList<C>(); listC.add(new C()); listC.add(new C());

           this.method1(listA); this.method1(listB); this.method1(listC);

           this.method2(listA,elem);  this.method2(listB,elem); this.method2(listC,elem);

       }
}
```

Please complete the most specific wildcard types for the class Amain methods (method1 and method2, the

Solution:

For method1:

- The method retrieves the first element from a list.

- It must accept a list of any type that is a subtype of A.

- Correct wildcard: ArrayList<? extends A>

For method2:

- The method adds an element of type C to a list.

- The list must allow C as an element, meaning it should be of type C or its superclasses.

- Correct wildcard: ArrayList<? super C>

Justification:

- ? extends A ensures that method1 works with lists containing elements that inherit from A while maintaini

- ? super C ensures that method2 can accept lists of C or its superclasses, allowing elements of type C to l

3 (3p). Is the following Java code correct? Please explain your answer.

```java
class A {
    protected int f1;
    static int s1=0;
    public A(int a)  { this.f1=a*s1; s1=s1+1; }
    static int getS()  { return s1; }
    int getS1(int x)  {return (x*getS());}
}
```

Solution:

Yes, the code is correct, but it has an important behavior that should be noted:

Explanation:

1. s1 is a static variable, meaning it is shared across all instances of A.
2. The constructor initializes f1 using s1, then increments s1.
3. Since s1 is initially 0, the first instance of A will have f1 = 0 * a = 0.
4. For subsequent instances, s1 will be increased, leading to different values of f1.
5. getS() returns s1, which reflects the latest global s1 value.
6. getS1(int x) multiplies x by s1, which may not be the same s1 used in the constructor.

Potential Issue:

- If the constructor is called multiple times, each new instance will use an incremented s1, affecting the valu
- This behavior is intentional but could be misleading in certain use cases.

Final Answer: The code is correct, but s1's modification inside the constructor affects subsequent instance