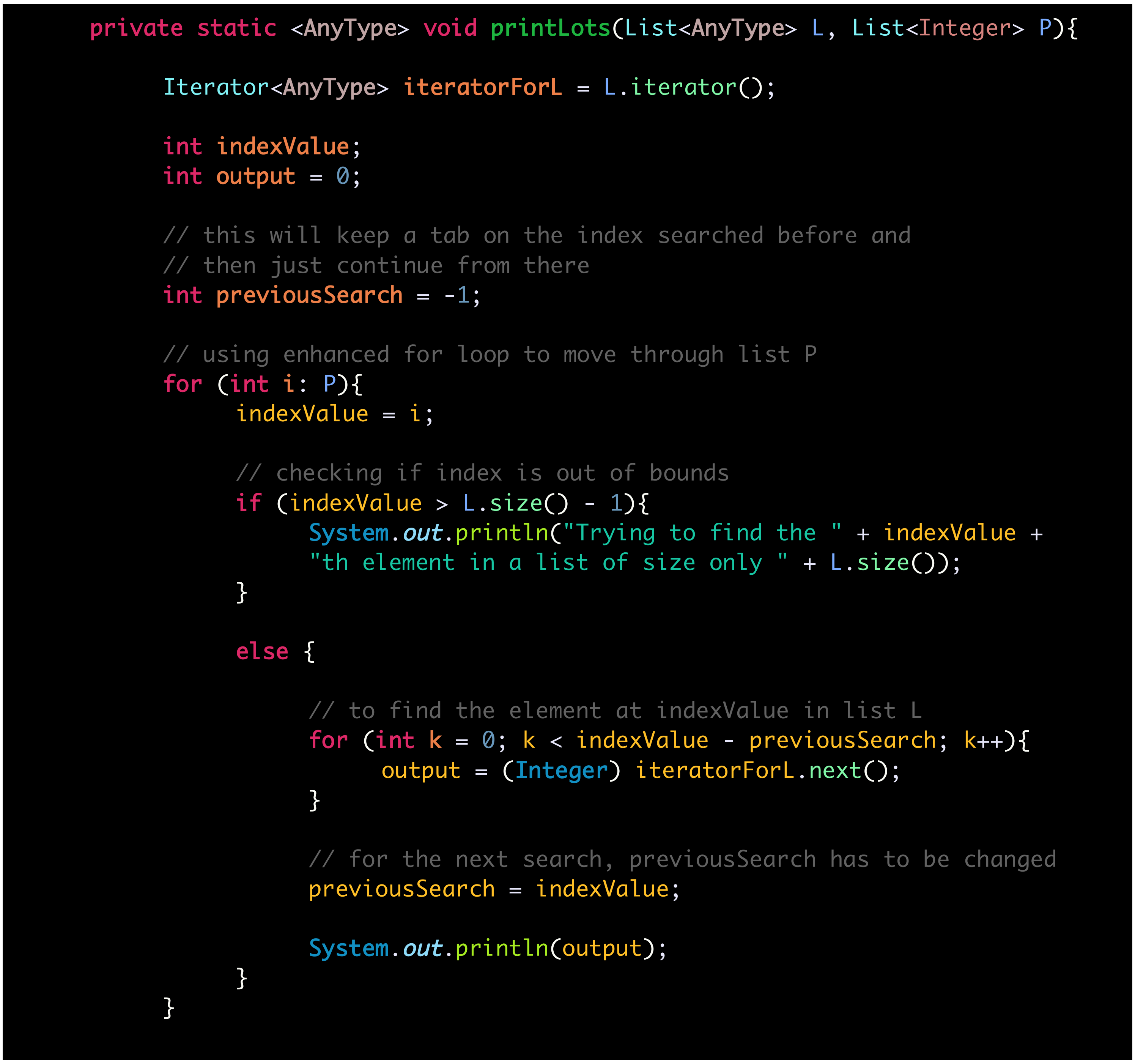
Problem 1



```java
private static <AnyType> void printLots(List<AnyType> L, List<Integer> P){

    Iterator<AnyType> iteratorForL = L.iterator();

    int indexValue;
    int output = 0;

    // this will keep a tab on the index searched before and
    // then just continue from there
    int previousSearch = -1;

    // using enhanced for loop to move through list P
    for (int i: P){
        indexValue = i;

        // checking if index is out of bounds
        if (indexValue > L.size() - 1){
            System.out.println("Trying to find the " + indexValue +
            "th element in a list of size only " + L.size());
        }

        else {

            // to find the element at indexValue in list L
            for (int k = 0; k < indexValue - previousSearch; k++){
                output = (Integer) iteratorForL.next();
            }

            // for the next search, previousSearch has to be changed
            previousSearch = indexValue;

            System.out.println(output);
        }
    }
```

Problem 2

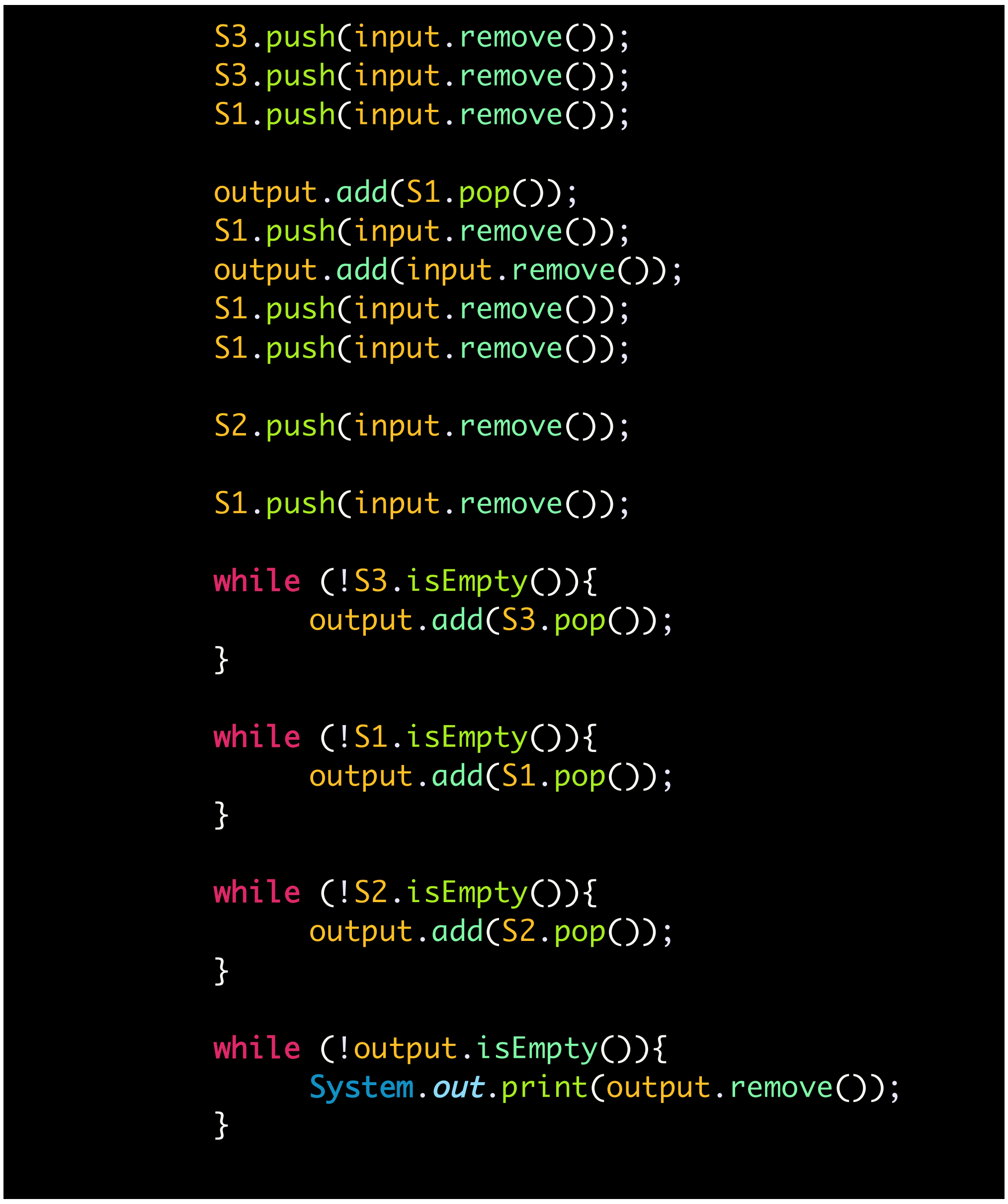
```
List intersection;
int start, end; (for L1)
int mid = (start + end / 2);

for (element in the list L2){

    if ( L1.midElement > element) {
        search to the left of the L1.midElement until L1.midElement ==
        element by changing midElement;
    }

    else if (L1.midElement < element){
        search to the right of the L1.midElement until L1.midElement ==
        element by changing mid_element;
    }

    else if (L1.midElement == element){
        add element to the list intersection;
    }

    When element has been found, change L1 such that only elements from
    the mid (the position at which element of L2 was found) to the end of L1
    are searched for the next element of L2
}
```

Problem 3

```
Array<AnyType> myArray = new Array[n];
int i = 0;
int k = n -1;

push1()
        myArray.add(i);
        i++;

push2()
        myArray.add(k);
        k--;

size1()
        return (i);

size2()
        return (n – k - 1);

size()
        return (size1() + size2());

pop1()
        return (myArray[--i]);
        if (size() == n): stack overflow

pop2()
        return (myArray[++k]);
        if (size() == n): stack overflow
```

Problem 4

(a)



```java
S3.push(input.remove());
S3.push(input.remove());
S1.push(input.remove());

output.add(S1.pop());
S1.push(input.remove());
output.add(input.remove());
S1.push(input.remove());
S1.push(input.remove());

S2.push(input.remove());

S1.push(input.remove());

while (!S3.isEmpty()){
        output.add(S3.pop());
}

while (!S1.isEmpty()){
        output.add(S1.pop());
}

while (!S2.isEmpty()){
        output.add(S2.pop());
}

while (!output.isEmpty()){
        System.out.print(output.remove());
}
```

(b) One example can be: 4, 9, 2, 7, 1, 6, 8, 3, 5