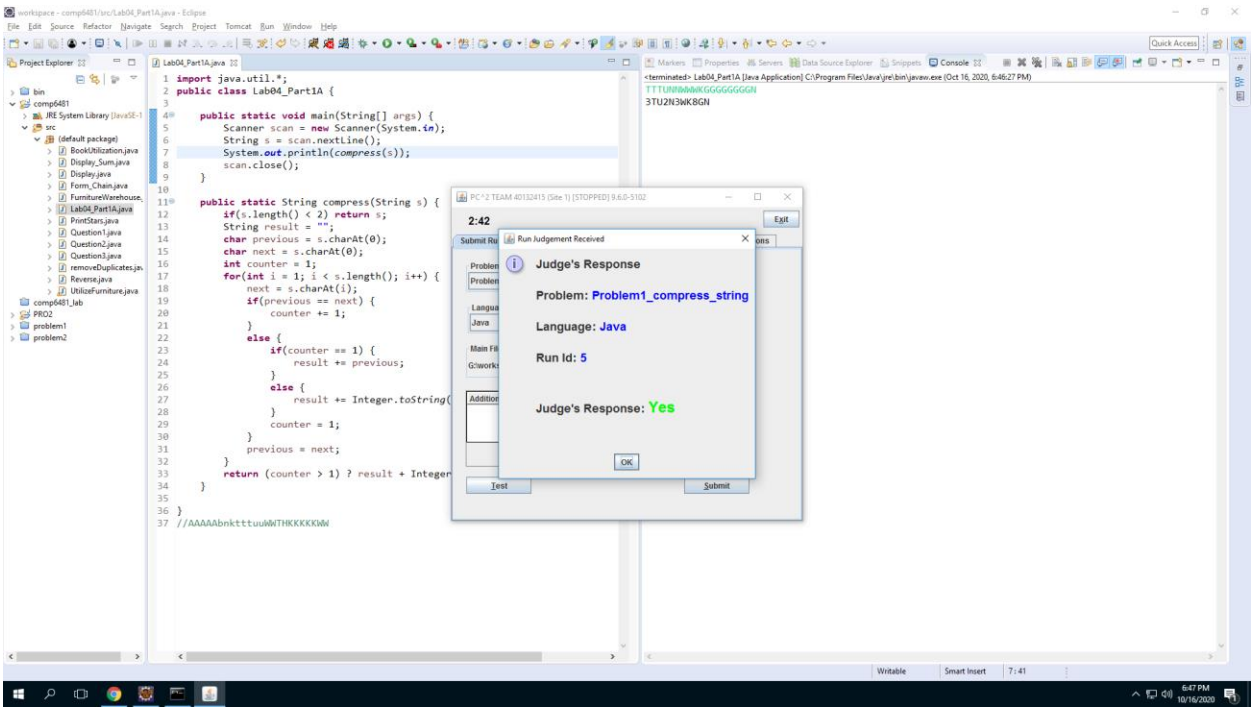
Problem 1A

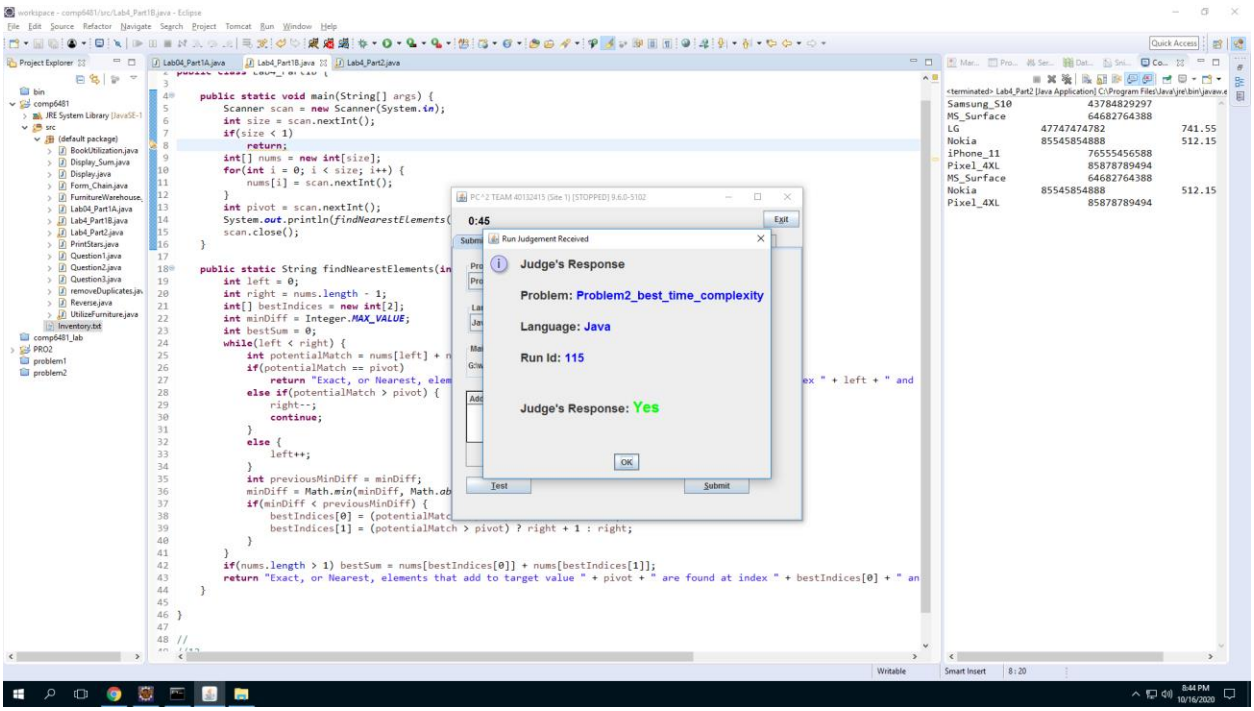
```
import java.util.*;
public class Lab04_Part1A {

	public static void main(String[] args) {
		Scanner scan = new Scanner(System.in);
		String s = scan.nextLine();
		System.out.println(compress(s));
		scan.close();
	}

	public static String compress(String s) {
		if(s.length() < 2) return s;
		String result = "";
		char previous = s.charAt(0);
		char next = s.charAt(0);
		int counter = 1;
		for(int i = 1; i < s.length(); i++) {
			next = s.charAt(i);
			if(previous == next) {
				counter += 1;
			}
			else {
				if(counter == 1) {
					result += previous;
				}
				else {
					result += Integer.toString(counter) +
previous;
				}
				counter = 1;
			}
			previous = next;
		}
		return (counter > 1) ? result + Integer.toString(counter) +
next : result + next;
	}

}
//Time: O(N) where N is the total length of the string as we traverse
the string exactly once.
//Space: O(1) as we do not count the resultant string in the space
complexity and we do not use any extra space.
```



Problem 1B

```java
import java.util.*;
public class Lab4_Part1B {

	@SuppressWarnings("resource")
	public static void main(String[] args) {
		Scanner scan = new Scanner(System.in);
		int size = scan.nextInt();
		if(size < 1)
			return;
		int[] nums = new int[size];
		for(int i = 0; i < size; i++) {
			nums[i] = scan.nextInt();
		}
		int pivot = scan.nextInt();
		System.out.println(findNearestElements(nums, pivot));
		scan.close();
	}

	public static String findNearestElements(int[] nums, int pivot) {
		int left = 0;
		int right = nums.length - 1;
```

```
            int[] bestIndices = new int[2];
            int minDiff = Integer.MAX_VALUE;
            int bestSum = 0;
            while(left < right) {
                  int potentialMatch = nums[left] + nums[right];
                  if(potentialMatch == pivot)
                        return "Exact, or Nearest, elements that add to
target value " + pivot + " are found at index " + left + " and index "
+ right + " adding to " + pivot;
                  else if(potentialMatch > pivot) {
                        right--;
                        continue;
                  }
                  else {
                        left++;
                  }
                  int previousMinDiff = minDiff;
                  minDiff = Math.min(minDiff, Math.abs(potentialMatch -
pivot));
                  if(minDiff < previousMinDiff) {
                        bestIndices[0] = (potentialMatch < pivot) ? left
- 1 : left;
                        bestIndices[1] = (potentialMatch > pivot) ? right
+ 1 : right;
                  }
            }
            if(nums.length > 1) bestSum = nums[bestIndices[0]] +
nums[bestIndices[1]];
            return "Exact, or Nearest, elements that add to target value
" + pivot + " are found at index " + bestIndices[0] + " and index " +
bestIndices[1] + " adding to " + bestSum;
      }

}

//Time O(N) as there may be two indices exactly in the center of the
array. In that case we would have traversed N - 1 elements which would
converge to N.
//Space: O(1) as we just a couple of pointers and variables to store
the sum and results.
```



Problem Part2

```
import java.util.*;
import java.io.*;
public class Lab4_Part2 {
	public static void main(String[] args) throws IOException{
		try {
			Scanner scan = null;
			PrintWriter pw = null;
			BufferedReader br = null;
			correct_Inventory(scan, pw);
			display_Good_Inventory(br);
		}catch(Exception e) {
			System.out.println(e);
		}
	}

	public static void correct_Inventory(Scanner sc, PrintWriter pw)
throws FileNotFoundException {
		try {
			sc = new Scanner(new
FileInputStream("G:\\workspace\\comp6481\\src\\Inventory.txt"));
			pw = new PrintWriter(new
FileOutputStream("G:\\workspace\\comp6481\\src\\Good_Inventory.txt"));
		}catch(Exception e) {
```

```
                System.out.println(e);
            }
            while(sc.hasNextLine()) {
                String title = "";
                long ID = 0;
                double price = 0;
                try {
                    title = sc.next();
                    ID = sc.nextLong();
                    price = sc.nextDouble();
                }catch(Exception e) {
                    System.out.println(e);
                }
                String temp = "";
                String endsWith = "";
                try {
                    temp = Long.toString(ID);
                    endsWith = temp.substring(temp.length() - 2,
temp.length());
                    if(endsWith.equals("33"))
                        continue;
                }catch(Exception e) {
                    System.out.println(e);
                }
                try {
                    if(endsWith.equals("99")) {
                        temp = temp.substring(0, temp.length() - 2)
+ "88";
                    }
                }catch(Exception e) {
                    System.out.println(e);
                }
                try {
                    pw.println(title + "\t\t" + Long.parseLong(temp)
+ "\t\t" + price);
                }catch(Exception e) {
                    System.out.println(e);
                }
            }
            sc.close();
            pw.close();
        }

        public static void display_Good_Inventory(BufferedReader br)
throws IOException {
            try {
```

```
                br = new BufferedReader(new
FileReader("G:\\workspace\\comp6481\\src\\Good_Inventory.txt"));

                String s = br.readLine();
                while(s != null) {
                      System.out.println(s);
                      s = br.readLine();
                }
          }catch(Exception e) {
                System.out.println(e);
          }
          br.close();
     }
}
```