Q1.

```
public class ReverseString {
   public static String reverse(String st){
      StringBuilder sb = new StringBuilder();
      StringBuilder result = new StringBuilder();
      int length = st.length();
      for(int i=0;i<length;i++){
         char ch = st.charAt(length -1 - i);
         if(ch == 32){
            result.insert(0, sb + " ");
            sb.setLength(0);
            continue;
         }
         sb.append(ch);
      }
      result.insert(0, sb + " ");

      return result.toString();
   }

   public static void main(String[] args){
      System.out.println(reverse("we test coders"));
   }
}
```

Running Time Calculation: O(n); where n is the length of the string.

Work	Time Complexity
creating string builder, calculating length, inserting and converting to string from string builder	O(c) + O(c) + O(c) = O(c)
Work inside loop like comparision, appending , indexing, length calculation	O(n) * (O(c) + O(c) + O(c) + O(c)) = O(n) * O(c) = O(n)

Total complexity = O(c) + O(n) => O(n)

Q2.

```
class MyStack {
  private final ArrayList<Integer> q;

  public MyStack() {
    this.q = new ArrayList();
  }

  public void push(int x){
   this.q.add(x);
  }

  public int pop(){
     return this.q.removeLast();
   }

  public int top(){
    return this.q.getLast();
  }

  public boolean empty(){
     return this.q.isEmpty();
  }
}
```

Q3.

```
class Solution {
     public ListNode reverseList(ListNode head){
       // only one item in the list
       if(head == null || head.next == null) return head;
       ListNode tail = null;
       while(head.next!=null){
          ListNode node = head.next;
         head.next = tail;
         tail = head;
        head = node;
      }
     head.next = tail;
    return head;
   }
}
```

Q4.

```
class Solution {
    public boolean isAnagram(String s, String t) {
    if(s.length() != t.length()) return false;

   HashMap<Character,Integer> map = new HashMap<>();
   for(int i=0;i<s.length();i++){
       if(map.containsKey(s.charAt(i))){
          map.put(s.charAt(i),map.get(s.charAt(i))+1);
       }else{
         map.put(s.charAt(i),1);
        }
    }

   for(int i=0;i<t.length();i++){
     Integer count = map.get(t.charAt(i));
     if(count==null) return false;
     if(count>1){
        map.put(t.charAt(i),count-1);
    }else{
       map.remove(t.charAt(i));
     }
   }
   return true;
  }
}
```