Question 1

(a)

(i)

5n^2+9n+7：

Order:5n^2>9n>7

does not include the low order terms :5n^2

the first term coefficients of this function：n^2

result：O(n^2)

(ii)

The time complexity is N^2,
i traverse from 0 to n, j traverse from current i to 0, the total number of times is n*(n-1)/2, ie complexity is n^2

(b)

```
public class StringQueueImpl implements StringQueue {
 private QueueCell head;
 private int size = 0;
 @Override
 public boolean isEmpty() {
        return size == 0;
 }

 @Override
 public void add(String c) {
        if (head == null) {
               head = new QueueCell(c);
        } else {
               QueueCell current = head;
               while (current != null) {
                      if (current.next == null) {
                             current.next = new QueueCell(c);
                             break;
                      } else {
                             current = head.next;
                      }
               }
        }
        size++;
 }
	@Override
	public String front() {
		if (size == 0 || head == null) {
			throw new QueueException();
		}
		return head.value;
	}

	@Override
	public void removeFront() {
		if (size == 0 || head == null) {
			throw new QueueException();
		}
		if (head.next != null) {
			head = head.next;
		} else {
			head = null;
		}
		size--;
	}

	class QueueCell {
		private QueueCell next;
		private String value;

		QueueCell(String str) {
			value = str;
		}
	}

	class QueueException extends RuntimeException {
		public QueueException() {
			super("Queue is null");
		}
	}
}
```

Question 2
(a)

non-computable : For a problem, no exact solution can be given.

If there is a set of functions B (that is, this set B is a set of functions f), and B is not an empty set, nor is it all functions f, then whether or not the random program P belongs to B is uncalculable.

(b)

In the initial state, the array to be sorted is divided into two parts that are sorted and unordered, and then traversal is performed, and the elements in the unsorted sequence are sequentially inserted into the sorted sequence.

the complexity:

The average complexity: the first step traverses the entire array n, then for each number, insert into the previous sorted array, you need to compare the sorted array n, that is, the complexity is n^2

The best complexity: When the element is an array, it only needs to traverse once, comparing only one number at a time. The complexity is n=n*1.

(c)

```
public int getNumOfNonLetters(BinaryTree<String> bt) {
    int num = 0;
    if (bt.isEmpty()) {
        return num;
    } else {
        num = getNumOfNonLetters(bt.leftChild())+1;
        num = num+getNumOfNonLetters(bt.rightChild());
    }
    return num;
}
```