문제 풀이 «더하기 (aplusb)»

부분문제 1

S가 두 글자이므로, 한 글자를 a에, 다른 글자를 b에 배정하는 두 가지 방법이 있습니다. 예로, 35를 $3+5$ 또는 $5+3$으로 나눌 수 있습니다. 두 방법 모두 합이 같으므로, 입력으로 주어진 숫자들의 합을 구하는 프로그램을 작성하면 됩니다.

```cpp
#include <iostream>
#include <string>

using namespace std;

int main() {
  int T; cin >> T;
  for(int t = 0; t < T; t++) {
    string S; cin >> S;
    cout << ((S[0] - '0') + (S[1] - '0')) << endl;
  }
}
```

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class Main {
  public static void main(String[] args) throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    int T = Integer.parseInt(br.readLine());
    for(int t = 0; t < T; t++) {
      String S = br.readLine();
      System.out.println(((S.charAt(0) - '0') + (S.charAt(1) - '0')));
    }
  }
}
```

```kotlin
fun main(args: Array<String>) {
  repeat(readLine()!!.toInt()) {
    println(readLine()!!.map{ it - '0' }.sum())
  }
}
```

부분문제 2

S가 세 글자인 경우를 생각하면, a와 b 둘 중 하나는 두 자리의 수이고, 다른 하나는 한 자리의 수일수밖에 없습니다. 즉, 가능한 경우는 $\overline{xy} + \overline{z}$ (x, y, z는 S의 글자들을 임의로 나열한 것) 꼴밖에 없고, 이 경우 합은 $10x + y + z$입니다. 따라서 가장 큰 숫자를 x의 위치에 두고, 다른 두 숫자는 아무렇게나 두면 됩니다.

```
fun main(args: Array<String>) {
  repeat(readLine()!!.toInt()) {
    val S = readLine()!!.map{ it - '0' }
    if(S.size == 2) {
      println(S.sum())
    }else if(S.size == 3) {
      println(S.max()!! * 9 + S.sum())
    }
  }
}
```

부분문제 3

주어진 숫자들을 a와 b 중 어떤 곳에 들어갈지 선택했다고 합시다. 예를 들어, S가 "235217"일 때, a에 '3', '2','7'이, b에 '2', '5', '1'이 들어간다는 것을 결정해 본 것입니다. 합을 최대화하고자 하므로, a와 b 둘 다 숫자들을 단조감소하도록 놓아야 합니다. 이 예시의 경우, $a = 732$, $b = 521$이어야 합니다.

각 숫자를 a나 b 둘 중 어느 곳에 들어갈 지 정하는 경우의 수는 $2^{|S|} - 2$가지 있습니다. 2를 빼는 이유는, 모든 숫자가 a 또는 b에 들어가는 경우를 배제하기 위해서입니다. 숫자들을 배정해 본 후, 각각을 내림차순 (정확히는 비오름차순) 정렬한 뒤 각각의 수를 구하고, 그 합을 구해 본 뒤, 이들 중 최댓값을 구하면 됩니다.

반복 횟수의 상한을 대략 $2^{|S|} \cdot |S| \log |S|$ 정도일 것으로 예상할 수 있습니다. 앞의 항은 모든 경우를 나열하기 때문에, 뒤의 항은 숫자들을 정렬하기 때문에 필요합니다. Counting Sort 등을 써서 $O(|S|)$로 줄일 수 있지만, 이 문제에서는 요구하지 않습니다.

```
fun main(args: Array<String>) {
  repeat(readLine()!!.toInt()) {
    val S = readLine()!!.map{ it - '0' }

    fun go(i: Int, a: List<Int>, b: List<Int>): Int
      = if(i == S.size) {
          if(!a.isEmpty() && !b.isEmpty()) {
            listOf(a, b).map{ it.sortedDescending().reduce { x, y -> 10 * x + y } }.sum
                ()
          }
          else {
            -1
          }
        }
        else {
          Math.max(go(i+1, a+S[i], b), go(i+1, a, b+S[i]))
        }

    println(go(0, listOf<Int>(), listOf<Int>()))
  }
}
```

부분문제 4

$|S| \le 16$이므로 $2^{|S|} \le 65\,536$이고, 이 정도면 충분히 작은 수이기 때문에 부분문제 3의 풀이와 같이 모든 경우를 일일이 순회해 보며 최댓값을 구하는 방법으로 풀 수 있습니다. 하지만 몇 가지 변화가 필요합니다.

- 16자리의 수가 주어지는데, C++/Java의 `int`형과 같은 32비트 부호 있는 정수형은 최대 $2^{31} \approx 2 \cdot 10^9$까지의 수만 저장할 수 있기 때문에, 답을 표현하기에 부적절합니다. C++의 `long long`형이나, Java의 `long`형을 사용하여 합을 저장해야 합니다.
- 몇 가지 아이디어를 통해 시간복잡도의 $|S|$ factor를 없앨 수 있습니다. 첫 번째로, 어차피 숫자들을 매번 내림차순 정렬하고 있으니, 그냥 시작부터 S를 내림차순 정렬해놓는 것입니다. 이렇게 하면 앞에서부터 숫자들을 a 또는 b에 차례대로 추가하면 정렬되어 있는 상태가 됩니다. 두 번째로, 어차피 숫자들이 정렬되어 있는 상태이니, 숫자들의 목록 대신 수 그 자체를 저장하는 것입니다. 예를 들어, `[5, 5, 3, 1, 1]` 대신 55311이라는 수를 저장할 수 있을 것입니다. 이 수의 맨 뒤에 새로운 숫자 '1'을 추가할 때에는, 10을 곱해서 왼쪽으로 한 칸 민 뒤 추가하고자 하는 숫자인 1을 더해주면 됩니다.

```
fun main(args: Array<String>) {
  repeat(readLine()!!.toInt()) {
    val S = readLine()!!.map{ (it - '0').toLong() }.sortedDescending()

    val n = S.reduce{ x, y -> 10L * x + y }

    if(n == 0L) {
      println(0)
    }else {
      fun go(i: Int, a: Long, b: Long): Long
        = if(i == S.size) (if(a + b < n) a + b else -1)
          else Math.max(go(i + 1, a * 10 + S[i], b), go(i + 1, a, b * 10 + S[i]))

      println(go(0, 0, 0))
    }
  }
}
```

부분문제 5

$|S| \leq 100$이므로 $2^{|S|} \approx 1.26 \times 10^{30}$이고, 모든 경우를 일일이 고려해 볼 수 없음을 알 수 있습니다.

부분문제 4의 프로그램으로 실험해보거나 직관을 통해서, a와 b 둘 중 하나는 무조건 한 자리 수임을 알 수 있습니다. 일반성을 잃지 않고 $a > b$라고 합시다.

직관적으로는, $a+b$의 자리수는 최대 max(a의 자리수, b의 자리수) $+1$이므로 자리수를 최대화할 필요가 있고, 이를 위해서는 a를 가능한 한 길게, b는 가능한 한 짧게 두는 것이 좋다고 생각할 수 있습니다.

엄밀한 증명은, b의 마지막 자리 숫자를 떼어 a의 마지막 자리에 붙여도 문제의 조건을 위배하지 않으면서 합은 증가한다는 사실을 이용하면 할 수 있습니다. $b = 10p+q$였다고 할 때, 원래 합은 $a+10p+q$였는데, b의 마지막 자리 숫자를 떼어내어 a에 붙이면 합은 $(10a+q)+p = a+10p+q+(9a-9p)$이고, $a > b \geq 10p > p$이므로 합이 무조건 증가합니다.

이 사실과 부분문제 3에서 봤던 '숫자를 비오름차순으로 배열해야 한다'는 성질을 이용하면, S를 비오름차순으로 정렬한 뒤 앞 $|S|-1$글자를 a로, 마지막 글자를 b로 두면 최적임을 알 수 있습니다. (a와 b의 마지막 글자끼리는 서로 바꿀 수 있기 때문에, 유일한 것은 아닙니다.)

답은 최대 100자리의 자연수이기 때문에 ('9'가 100개 주어지는 경우), `int`, `long`, `__int128` 등의 자료형으로는 답을 담을 수 없습니다. 이를 피하기 위한 두 가지 방법이 있습니다.

- C++의 경우: 긴 자리 수의 덧셈을 직접 구현합니다. 방법을 설명하자면, 먼저 a를 길이가 $|S|-1$인 배열에 한 자리씩 담아 표현한 뒤, 맨 뒷 자리에 b를 더합니다. 맨 뒷 자리의 값이 10 미만이면 덧셈이 잘 된 것이고, 10 이상이면 받아올림을 했다는 표시를 한 뒤 앞 자리로 이동하여 같은 과정을 반복합니다. 자세한 방법은 코드를 확인하실 수도 있겠으나, 본인이 긴 자리 수와 한 자리 수를 손으로 더할 때 사용하시는 방법을 절차화하여 코드로 옮겨 보는 것을 추천합니다.
- Java의 경우: C++과 같이 덧셈을 직접 구현하거나, `java.math.BigInteger`를 사용합니다.

실수할 수 있는 데이터로 모든 숫자가 9인 경우(자리수가 하나 늘어납니다)가 있습니다.

```
#include <iostream>
#include <string>
#include <vector>

using namespace std;

int main() {
  int T; cin >> T;
  for(int t = 0; t < T; t++) {
    string S; cin >> S;
    vector<int> a; for(char c : S) a.push_back(c - '0');

    for(int i = 0; i < a.size(); i++) {
      for(int j = i+1; j < a.size(); j++) {
        if(a[i] < a[j]) {
          int tmp = a[i];
          a[i] = a[j];
          a[j] = tmp;
        }
      }
    }

    int b = a.back();
    a.pop_back();
    for(int i = int(a.size()) - 1; i >= 0; i--) {
      a[i] += b;
      b = 0;
      if(a[i] < 10) break;
      a[i] -= 10;
      b = 1;
    }
```

```
    if(b == 1) {
      cout << '1';
    }
    for(int i = 0; i < int(a.size()); i++) {
      cout << a[i];
    }
    cout << endl;
  }
  return 0;
}
```

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.util.Arrays;
import java.util.Collections;
import java.math.BigInteger;

public class Main {
  public static void main(String[] args) throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    int T = Integer.parseInt(br.readLine());
    for(int t = 0; t < T; t++) {
      char[] S = br.readLine().toCharArray();
      Arrays.sort(S);
      String X = new StringBuilder(new String(S)).reverse().toString();
      BigInteger a = new BigInteger(X.substring(0, X.length() - 1));
      BigInteger b = new BigInteger(X.substring(X.length() - 1, X.length()));
      System.out.println(a.add(b));
    }
  }
}
```

```
import java.math.BigInteger
fun main(args: Array<String>) {
  repeat(readLine()!!.toInt()) {
    val s = readLine()!!.toList().sortedDescending().joinToString("")
    println(BigInteger(s.dropLast(1)) + BigInteger(s.takeLast(1)))
  }
}
```

부분문제 6

부분문제 5와 풀이가 완전히 똑같으나, 다음과 같은 요소들로 인해 틀리거나 제한 시간 내에 답을 구하지 못 할 수 있습니다.

- S를 비내림차순 정렬할 때, Insertion sort, Selection sort 등과 같이 시간복잡도가 $O(|S|^2)$인 것을 사용하면 안 됩니다. 정렬로 인한 반복 횟수가 대략 $T \cdot |S|^2$회라고 생각할 수 있는데, 이는 최대 $1\,000 \cdot 3\,000^2 = 9 \times 10^9$로 2초 안에 프로그램이 종료할 것이라고 보기 어렵습니다.
- Java의 `java.math.BigInteger`를 사용하면 제한 시간 내에 답을 구할 수 없습니다. (의도한 것입니다.) 덧셈을 직접 구현하셔야 합니다.
- Java에서 `String`에 `+=` 연산을 적용하여 한 글자씩 더하여 정답을 저장하면 시간복잡도가 $O(|S|^2)$이 됩니다. `+=` 연산을 할 때마다 새로운 `String`이 생성되기 때문입니다. `StringBuilder` 등을 사용하여 문자열을 만든 뒤 출력하는 방법을 권장합니다.

```cpp
#include <iostream>
#include <string>
#include <vector>

using namespace std;

int main() {
  int T; cin >> T;
  for(int t = 0; t < T; t++) {
    string S; cin >> S;

    vector<int> cnt(10);
    for(char c : S) cnt[c - '0'] += 1;

    vector<int> a;
    for(int d = 9; d >= 0; d--) {
      for(int rep = 0; rep < cnt[d]; rep++) {
        a.push_back(d);
      }
    }

    int b = a.back();
    a.pop_back();
    for(int i = int(a.size()) - 1; i >= 0; i--) {
      a[i] += b;
      b = 0;
      if(a[i] < 10) break;
      a[i] -= 10;
      b = 1;
    }

    if(b == 1) {
      cout << '1';
    }
    for(int i = 0; i < int(a.size()); i++) {
      cout << a[i];
    }
    cout << endl;
  }
  return 0;
}
```

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.util.Arrays;
```

```java
public class Main {
  public static void main(String[] args) throws IOException {
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    int T = Integer.parseInt(br.readLine());
    for(int t = 0; t < T; t++) {
      String S = br.readLine();
      int[] a = new int[S.length()];
      for(int i = 0; i < S.length(); i++) {
        a[i] = S.charAt(i) - '0';
      }
      Arrays.sort(a);
      for(int i = 0; i < a.length - i - 1; i++) {
        int j = a.length - i - 1;
        int tmp = a[i];
        a[i] = a[j];
        a[j] = tmp;
      }

      int b = a[a.length - 1];
      for(int i = a.length - 2; i >= 0; i--) {
        a[i] += b;
        b = 0;
        if(a[i] < 10) {
          break;
        }
        a[i] -= 10;
        b = 1;
      }

      StringBuilder result = new StringBuilder();
      if(b == 1) {
        result.append("1");
      }
      for(int i = 0; i < a.length - 1; i++) {
        result.append(String.valueOf(a[i]));
      }
      System.out.println(result);
    }
  }
}
```

```kotlin
fun main(args: Array<String>) {
  repeat(readLine()!!.toInt()) {
    val s = readLine()!!.map{ it - '0' }.sortedDescending()
    val a = s.dropLast(1).toIntArray()
    var b = s.last()

    for(i in a.lastIndex downTo 0) {
      a[i] += b
      b = 0
      if(a[i] < 10) break
      a[i] -= 10
      b = 1
    }

    if(b == 1) print(1)
    println(a.joinToString(""))
  }
}
```