Lab5 Report

1. Algorithm explanation

First, we need to read the input. We will getc twice and check if the second is enter, which can help us judge if the first number is less than 10 or not. Then we need to read the following 2n numbers. Each time we read a char we will getc twice and check if the second getc get a letter or not. Finally, we will store all the number in memory from x3100.After the input, r1 will contain the number of the counting number int the first line.

Second, we will call function find twice. In one time we will find from the left number in second line. In one time we will find from the right number in the second line.

Third, we will get the output sequence in memory from x3128 since the input must have at least one solution.

Fourth, in the function find, we need two parameters, the memory r1 point to is a number in input sequence, the memory point to is the empty element in output sequence, find function will try to store *r1 in r2 after checking if *r1 has never been stored in the output sequence. After we go into the function, we will first check if the output sequence is full. We will return 1 if it is full. If the storing is rejected, find will return 0. If the storing is okay, then find function will call find to try to store the left element in the next line and the right number in the next line. And find will return 1 if one of the two calls return 1. R5 is used to store the return number.

2. Essential codes

```
81          lea r1,storage
82          lea r2,storeout
83          and r5,r5,#0
84          jsr find    ;try if we first put the left number in the first line
85          add r5,r5,#0
86          brp skip3
87          add r1,r1,#1
88          jsr find    ;try if we first put the right numbe in the first line
```

The two calls in second part

```
130         ld r3,number    ;check if the numbers in the sequence is enough
131         lea r4,storeout
132         add r3,r3,r4
133         not r3,r3
134         add r3,r3,#1
135         add r3,r3,r2
136         brz ret1
```

R3 is the total number. R4 is the beginning address of the output sequence, after adding them up we need to compare it with r2, the address we want to store next, to see if the output sequence is full.

```
152 canst   str r4,r2,#0    ;store *r1 and try next to store numbers in next line
153         add r2,r2,#1
154         and r3,r1,#1
155         brz add2        ;add1 branch means we can get next line by adding 1 to r1
156         add r1,r1,#1
157         jsr find
158         add r5,r5,#0
159         brp ret1
160         add r1,r1,#1
161         jsr find
162         brp ret1
163         brnzp ret0
```

Because the input is stored from x3100. So we have stored the left number if r1 is even, and we have store the right number if r1 is odd. If r1 is odd we need to add 1 or 2 to let r1 point to the left and right number in next line. If r1 is even we need to add 2o or 3 to let r1 point to the left and right number in the next line. If the next call returns 1, it will return 1 at once,

3. TA question

Actually, there is no extra questions. TA just let me explain my algorithm.