Lab5-Report

Algorithm Explanation

Firstly,we need some stacks. One to store R7 and state now,two to store the two input list.

```
            LD R4,STACK1    ;;R5 <- x4000
            LD R5,STACK2    ;;R5 <- x5000
            LD R6,RETSTACK  ;;R6 <- xFDFF
```

Then we should do with input: we should read N and read n rows one by one. Put them into two stacks: This is READN (read the N).

```
;;Decimal Input N -> R1
READN       TRAP x20
            TRAP x21
            ADD R2,R0,#-10  ;;(R2 <- R0ASC - 10)
            BRz NEXT
            JSR Multi10     ;;(R1 = R1 * 10)
            LD  R3,ASC-48   ;;(R3 <- -48)
            ADD R0,R0,R3    ;;(R0 = R0 - 48)
            ADD R1,R1,R0    ;;(R1 = R1 + R0)
            BRnzp   READN
```

As well as before, we read the n rows one by one.

Then we write this recursive algorithm. we choose list 1 and go deeper , and choose list 2 and go deeper. When it is deepest,we judge if it is true. If not,we return the last recursion and choose 2 and go on:

```
SEARCH      ADD R6,R6,#-1   ;;R6 <- R6 - 1
            STR R7,R6,0     ;;MEM[R6] <- R7
            ADD R6,R6,#-1   ;;R6 <- R6 - 1
            ADD R4,R4,#-1   ;;R4 <- R4 - 1
            ADD R5,R5,#-1   ;;R5 <- R5 - 1
            ADD R3,R3,#1    ;;R3 <- R3 + 1

            BR  Choose1     ;;choose stack1
Choose1back JSR SEARCH      ;;go deeper
            BR  Choose2     ;;in Choose2 ,one more thing:judge if it is deepest.
Choose2back JSR SEARCH      ;;go deeper
Deepest     BR  Judge
Continue    ADD R3,R3,#-1   ;;R3 <- R3 - 1
            ADD R4,R4,#1    ;;R4 <- R4 + 1
            ADD R5,R5,#1    ;;R5 <- R5 + 1
            ADD R6,R6,#1    ;;R6 <- R6 + 1
            LDR R7,R6,0     ;;R7 <- MEM[R6]
            ADD R6,R6,#1    ;;R6 <- R6 + 1

SEARCHRET   RET
```

When it comes to judging if it is true, I map the N number into corresponding CONST bit number. For example, 5 map to 0000 0000 0011 1110; 2 map to 0000 0000 0000 0110. Then at every choose step, we will get this number into state now:(For example 5 map to 0000 0000 0010 0000; 2 map to 0000 0000 0000 0100).

```
MAPTOBIT1   LDR R2,R6,#2       ;;R2 <- MEM[R6] (BITNUMBER)
            LDR R1,R3,0         ;;R1 <- MEM[R3] (counter) (;;R0 <- BIT[R1]:1)
            AND R0,R0,0         ;;R0 <- 0
            ADD R0,R0,#1        ;;R0 <- R0 + 1
Loop1        ADD R0,R0,R0        ;;R0 << 1
            ADD R1,R1,#-1       ;;R1 <- R1 - 1 (counter --)
            BRp Loop1

            ADD R1,R2,0         ;;R1 <- R2(BITNUMBER)
            NOT R0,R0
            NOT R1,R1
            AND R0,R0,R1
            NOT R0,R0           ;;R0 <- R0 | R1

            ADD R2,R0,0         ;;R2(BITNUMBER) <- R0
            STI R2,BITNUMBER    ;;MEM[x6003] <- R2
            STR R2,R6,0         ;;MEM[R6] <- R2

            BR BACK1
```

And finally we judge it: (whether state now is equal to BITCONST)

```
Judge       LDI R0,BITNUMBER
            LDI R1,BITCONST
            AND R0,R0,R1        ;;R0 <- R0 & R1
            NOT R0,R0
            ADD R0,R0,#1        ;;R0 <- -R0
            ADD R0,R1,R0        ;;R0 <- R1 - R0
            BRz FINISH
            BR  Continue
```

What's more, we should pay attention to the decimal output:

```
FINISH      LDI R1,CONSTNUM     ;;R1 <- N
            LD R2,LIST          ;;R2 <- x7000
            ADD R1,R2,R1        ;;R1 <- R1 + R2
            ADD R1,R1,#1        ;;R1 <- R1 + 1
            AND R0,R0,0         ;;R0 <- 0
            STR R0,R1,0         ;;MEM[R1] <-0           (The end hint :0)
            LD R1,LIST          ;;R1 <- x7000
Nextout     ADD R1,R1,#1        ;;R1 <- R1 + 1
            LDR R2,R1,0         ;;R2 <- MEM[R1](R2 : Decimal number)
            BRz Outover
            ADD R0,R2,#-9       ;;R0 <- R2(Decimal number) -9
            BRp Twooutput
            LD  R0,ASC0         ;;R0 <- 48
            ADD R0,R2,R0        ;;R0 <- R2 + 48
            TRAP x21
            LD R0,ASCSP
```

```
            TRAP x21
            BR Nextout
Twooutput   LD R0,ASC1          ;;R0 <- ASC1
            TRAP x21
            ADD R0,R2,#-9       ;;R0 <- R2(Decimal number) -9
            ADD R0,R0,#-1       ;;R0 <- R0 - 1
            LD  R2,ASC0         ;;R2 <- 48
            ADD R0,R0,R2        ;;R0 <- R0 + 48
            TRAP x21
            LD R0,ASCSP
            TRAP x21
            BR  Nextout
Outover     HALT
```

Then we finished!

Answer & Question

1. How do you judge if it is true?

   Answer: judge if the state now is equal to BITCONST.(We change every number to corrsponding bit already)

2. How do you store now state?

   Answer: Push it into stack together with R7.