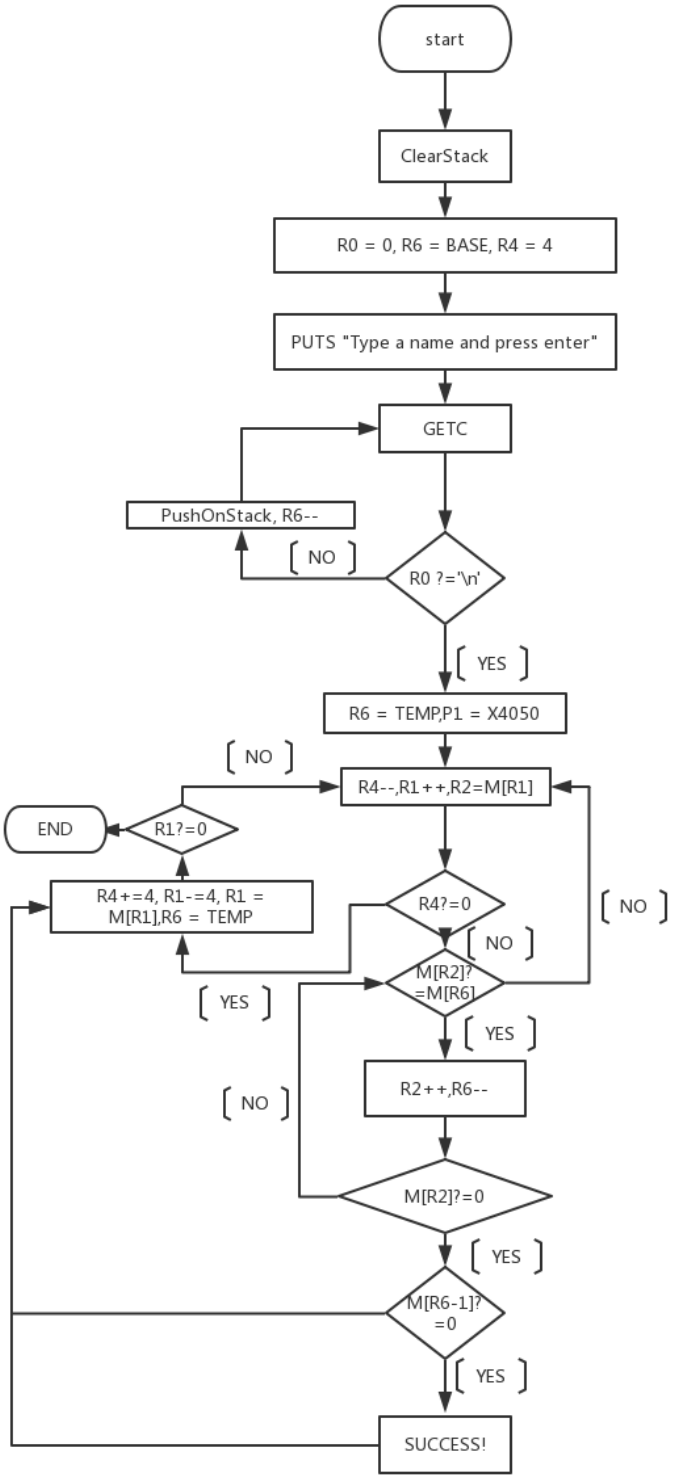
The 2nd lab report

(Due on Jul.13th)

1 .Program algorithm:



3.Brief Explanations

STEP1-INITIALIZATION: Firstly, we should do some initialization. Clear the stack at the very beginning. Then set R0, which serves as returning value, with value 0, then store value 0 in to SAVE, which serves as a flag of output. If SAVE is loaded with value 0 at the end of program, it will output "No entry.". Then LEA R0 with prompt and print it. Initialize Stack-pointer R6 with the base(TEMP) of the stack which is used to store the input string. Finally, initialize R4 with value 4. It serves as the counter of elements in a node (4 elements).

STEP2-GETSTR: After initialization, we should get a string and push it into stack. Characters are input one by one, until a x20(\n) is input. When inputting, we invoking the PUSH-Function by using the instruction JSR. It is vital to pay attention to the value of R7.

STEP3-FOUND1: firstly, let pointer R6 go down to the base of stack (LEA R6, TEMP). I have to admit that it's not the typical way to traversing a stack and popping the value out, but this way is much more convenient in this case.

During the whole procedure of searching, R1 serves as a second-rank pointer(PTR1) and R2 serves as a first-rank pointer(PTR2). R1 points to the element of node, while R2 points to every character of an element. Then move R1 and R2 to compare every character with the input character stored on the stack.

R1 will move to the next node when the counter R4 has reduced to 0. R2 will return to the first element of a node when it reaches the end of a string(x0000).

STEP4-FOUND2: When characters are all the same, we must make sure that the pointers of them are both at the x0000. If so, the program will enter the FOUND2 step and print every element of the corresponding node. Then return to the FOUND1(search) step.

The whole program will halt when R1 equals to x0000.

3.Source Code (in appendix)