```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAX 10  // Maximum size for arrays

// Global variables to store productions and sets
char prod[MAX][MAX], first[MAX][MAX], follow[MAX][MAX];
int n;  // Number of productions

// Function to add a character to a set if it doesn't already exist
void addToSet(char set[], char val) {
    if (!strchr(set, val)) strncat(set, &val, 1);
}

// Function to compute the FIRST set of a non-terminal
void findFirst(char c, char firstSet[]) {
    // If c is a terminal, add it to the FIRST set and return
    if (!isupper(c)) { addToSet(firstSet, c); return; }

    // If c is a non-terminal, find its FIRST set
    for (int i = 0; i < n; i++) {
        if (prod[i][0] == c) {
            // Recursively find FIRST set of the first symbol in the production's right-hand side
            findFirst(prod[i][2], firstSet);
        }
    }
}

// Function to compute the FOLLOW set of a non-terminal
void findFollow(char c, char followSet[]) {
    // If c is the start symbol, add $ to its FOLLOW set
    if (c == prod[0][0]) addToSet(followSet, '$');

    // Look for c in the right-hand side of all productions
    for (int i = 0; i < n; i++) {
        for (int j = 2; prod[i][j] != '\0'; j++) {
            if (prod[i][j] == c) {
                // If c is followed by another symbol
                if (prod[i][j + 1] != '\0') {
                    // Add the FIRST of the following symbol to FOLLOW(c)
                    char temp[MAX] = "";
                    findFirst(prod[i][j + 1], temp);
                    for (int k = 0; temp[k] != '\0'; k++) addToSet(followSet, temp[k]);
                } else {
                    // If c is at the end of a production, add FOLLOW of the left-hand side to FOLLOW(c)
                    findFollow(prod[i][0], followSet);
                }
            }
        }
    }
}

int main() {
    // Get user input for the grammar
    printf("Enter number of productions: ");
    scanf("%d", &n);
    printf("Enter productions (Format: A=α):\n");
    for (int i = 0; i < n; i++) scanf("%s", prod[i]);

    // Compute FIRST and FOLLOW sets for each non-terminal
    for (int i = 0; i < n; i++) {
        first[i][0] = follow[i][0] = '\0';  // Initialize sets as empty
        findFirst(prod[i][0], first[i]);    // Compute FIRST set
        findFollow(prod[i][0], follow[i]);  // Compute FOLLOW set
    }

    // Display the results
    printf("\nFIRST sets:\n");
    for (int i = 0; i < n; i++) printf("FIRST(%c) = { %s }\n", prod[i][0], first[i]);

    printf("\nFOLLOW sets:\n");
    for (int i = 0; i < n; i++) printf("FOLLOW(%c) = { %s }\n", prod[i][0], follow[i]);

    return 0;
}
```