2nd Normal form(detection and splitting if needed)
Code written in C.

Code:

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

#define MAX_ATTR 6
#define MAX_FD 4
#define MAX_LEN 3

char attributes[MAX_ATTR][MAX_LEN];
char left[MAX_FD][MAX_LEN];
char right[MAX_FD][MAX_LEN];
char candidate[MAX_ATTR];
int nattr, nfd;
int found[26] = {0};

void readInput() {
    printf("Enter number of attributes: ");
    scanf("%d", &nattr);
    printf("Enter attributes (space separated): ");
    for (int i = 0; i < nattr; i++) {
        scanf("%s", attributes[i]);
        if (isalpha(attributes[i][0]))
            found[tolower(attributes[i][0]) - 'a'] = 1;
    }
    printf("Enter number of functional dependencies: ");
    scanf("%d", &nfd);
    getchar();  // Consume the newline character left by scanf
    for (int i = 0; i < nfd; i++) {
        printf("Enter the functional dependencies LHS for %d: ", i);
        scanf("%s", left[i]);
    }
    for (int i = 0; i < nfd; i++) {
        printf("Enter the functional dependencies RHS for %d: ", i);
        scanf("%s", right[i]);
    }
    printf("Enter the candidate key: ");
    scanf("%s", candidate);
}
```

```
int isSubset(const char *set, const char *subset) {
    for (int i = 0; subset[i] != '\0'; i++) {
        if (!strchr(set, subset[i])) {
            return 0; // Not a subset
        }
    }
    return 1; // Is a subset
}

int Set(const char *s1, const char *s2) {
    if (strlen(s1) != strlen(s2))
        return 0;
    for (int i = 0; s1[i] != '\0'; i++) {
        if (strchr(s2, s1[i]) == NULL)
            return 0;
    }
    return 1;
}


void resolve(const char * attributes,int j){
    char s[MAX_ATTR]="";
    for(int i=0;attributes[i]!='\0';i++){
        if(attributes[i]!=right[j][0]){
            char a[2] = {attributes[i], '\0'};
            strcat(s,a);
        }
    }
    printf("Subset is (%s%s)\n",left[j],right[j]);
    if(is2nf(s))
    printf("Subset is (%s)\n",s);
}

int is2nf(const char *attributes) {
    char nonPrime[MAX_ATTR] = "";
    for (int i = 0; attributes[i] != '\0'; i++) {
        if (strchr(candidate, attributes[i]) == NULL) {
            char s[2] = {attributes[i], '\0'};
            strcat(nonPrime, s);
        }
    }
    for (int i = 0; nonPrime[i] != '\0'; i++) {
        for (int j = 0; j < nfd; j++) {
            if (right[j][0] == nonPrime[i]) {
```

```
            if (!Set(left[j], candidate) && isSubset(candidate, left[j])) {
                printf("Not in 2NF\n");
                resolve(attributes,j);
                return 0;
            }
        }
    }
    return 1;
}

int main() {
    readInput();
    char attr[MAX_ATTR * MAX_LEN] = ""; // Create a sufficiently large array
    for (int i = 0; i < nattr; i++) {
        strcat(attr, attributes[i]);
    }
    if (is2nf(attr)) {
        printf("In 2NF\n");
    }
    return 0;
}
```

Output:

```
reethi@DESKTOP-8744EFO:~/dir1/dbms$ ./a.out
Enter number of attributes: 6
Enter attributes (space separated): s u h e n l
Enter number of functional dependencies: 4
Enter the functional dependencies LHS for 0: su
Enter the functional dependencies LHS for 1: s
Enter the functional dependencies LHS for 2: u
Enter the functional dependencies LHS for 3: u
Enter the functional dependencies RHS for 0: h
Enter the functional dependencies RHS for 1: e
Enter the functional dependencies RHS for 2: n
Enter the functional dependencies RHS for 3: l
Enter the candidate key: su
Not in 2NF
Subset is (se)
Not in 2NF
Subset is (un)
Not in 2NF
Subset is (ul)
```

Subset is (suh)