3rd Normal form(detection and splitting if needed)
Code written in C.(Algorithm as provided in the standard reference book)

Code:

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

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

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() {
    int mc;
    char left_1[10][MAX_LEN];
    char right_1[10][MAX_LEN];
    printf("Enter the number of fds in canonical cover: ");
    scanf("%d", &mc);
    for (int i = 0; i < mc; i++) {
        printf("Enter the functional dependencies LHS for %d: ", i);
        scanf("%s", left_1[i]);
    }
    for (int i = 0; i < mc; i++) {
        printf("Enter the functional dependencies RHS for %d: ", i);
        scanf("%s", right_1[i]);
    }
    int i = 0;
    char* result[10] = {NULL}; // Initialize to NULL
    for (int j = 0; j < mc; j++) {
        int yes = 0;
        char temp[MAX_LEN*2];
        snprintf(temp, sizeof(temp), "%s%s", left_1[j], right_1[j]);
        for (int k = 0; k < i; k++) {
            if (isSubset(result[k], temp)) {
                yes = 1;
                break;
            }
```

```
        }
        if (yes == 0) {
            result[i++] = strdup(temp);
        }
    }
    int yes = 0;
    for (int k = 0; k < i; k++) {
        if (isSubset(result[k], candidate)) {
            yes = 1;
            break;
        }
    }
    if (yes == 0) {
        result[i++] = strdup(candidate);
    }
    printf("Final results:\n");
    for (int k = 0; k < i; k++) {
        printf("%s\n", result[k]);
        free(result[k]);
    }
    return;
}

int is3nf(const char *attributes) {
    for(int i=0;i<nfd;i++){
        if(isSubset(left[i],right[i]) || (Set(left[i],candidate))){
            continue;
        }
        char s[MAX_LEN]="";
        for(int j=0;right[i][j]!='\0';j++){
            if(strchr(left[i],right[i][j])==NULL){
                char a[2] = {right[i][j], '\0'};
                strcat(s,a);
            }
        }
        if(isSubset(candidate,s))
        continue;
        else
        {
            printf("Not in 3NF\n");
            resolve();
            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 (is3nf(attr)) {
        printf("In 3NF\n");
    }
    return 0;
}
```

Output:

```
reethi@DESKTOP-8744EFO:~/dir1/dbms$ ./a.out
Enter number of attributes: 5
Enter attributes (space separated): o n s c a
Enter number of functional dependencies: 4
Enter the functional dependencies LHS for 0: o
Enter the functional dependencies LHS for 1: o
Enter the functional dependencies LHS for 2: s
Enter the functional dependencies LHS for 3: o
Enter the functional dependencies RHS for 0: n
Enter the functional dependencies RHS for 1: s
Enter the functional dependencies RHS for 2: c
Enter the functional dependencies RHS for 3: a
Enter the candidate key: o
Not in 3NF
Enter the number of fds in canonical cover: 4
Enter the functional dependencies LHS for 0: o
Enter the functional dependencies LHS for 1: o
Enter the functional dependencies LHS for 2: s
Enter the functional dependencies LHS for 3: o
Enter the functional dependencies RHS for 0: n
Enter the functional dependencies RHS for 1: s
Enter the functional dependencies RHS for 2: c
Enter the functional dependencies RHS for 3: a
Final results:
on
os
sc
```

oa