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

void grammarfollow(char, int, int);
void follow(char c);
void find_first(char, int, int);
int count, n = 0;
char final_first[10][100];
char final_follow[10][100];
int m = 0;

char production[10][10];
char f[10], first[10];
int k;
char ck;
int e;

int main(int argc, char** argv)
{
    int jm = 0;
    int km = 0;
    int i, choice;
    char c, ch;
    count = 3;
    strcpy(production[0], "E=AB");
    strcpy(production[1], "A=ilove");
    strcpy(production[2], "B=jtptutorials");
    int ff;
    char done[count];
    int ptr = -1;
    for (k = 0; k < count; k++) {
        for (ff= 0; ff< 100; ff++) {
            final_first[k][ff] = '!';
        }
    }
    int point1 = 0, point2, xxx;
    for (k = 0; k < count; k++) {
        c = production[k][0];
        point2 = 0;
```

```
        xxx = 0;
        for (ff= 0; ff<= ptr; ff++)
            if (c == done[ff])
                xxx = 1;

        if (xxx == 1)
            continue;

        // Calling function
        find_first(c, 0, 0);
        ptr += 1;
        done[ptr] = c;
        printf("\n First(%c) = { ", c);
        final_first[point1][point2++] = c;
        for (i = 0 + jm; i < n; i++) {
            int fs = 0, chk = 0;
            for (fs = 0; fs< point2; fs++) {
                if (first[i] == final_first[point1][fs]) {
                    chk = 1;
                    break;
                }
            }
            if (chk == 0) {
                printf("%c, ", first[i]);
                final_first[point1][point2++] = first[i];
            }
        }
        printf("}\n");
        jm = n;
        point1++;
    }
    printf("\n");
    printf("================================""\n\n");
    char donee[count];
    ptr = -1;
    for (k = 0; k < count; k++) {
        for (ff= 0; ff< 100; ff++) {
            final_follow[k][ff] = '!';
        }
    }
```

```
    point1 = 0;
    int land = 0;
    for (e = 0; e < count; e++) {
        ck = production[e][0];
        point2 = 0;
        xxx = 0;
        for (ff= 0; ff<= ptr; ff++)
            if (ck == donee[ff])
                xxx = 1;

        if (xxx == 1)
            continue;
        land += 1;
        follow(ck);
        ptr += 1;
        donee[ptr] = ck;
        printf(" Follow(%c) = { ", ck);
        final_follow[point1][point2++] = ck;
        for (i = 0 + km; i < m; i++) {
            int fs= 0, chk = 0;
            for (fs= 0; fs< point2; fs++) {
                if (f[i] == final_follow[point1][fs]) {
                    chk = 1;
                    break;
                }
            }
            if (chk == 0) {
                printf("%c, ", f[i]);
                final_follow[point1][point2++] = f[i];
            }
        }
        printf(" }\n\n");
        km = m;
        point1++;
    }
}

void follow(char c)
{
    int i, j;
```

```
    if (production[0][0] == c) {
        f[m++] = '$';
    }
    for (i = 0; i < 10; i++) {
        for (j = 2; j < 10; j++) {
            if (production[i][j] == c) {
                if (production[i][j + 1] != '\0') {
                    grammarfollow(production[i][j + 1], i,
                            (j + 2));
                }

                if (production[i][j + 1] == '\0'
                        && c != production[i][0]) {
                    follow(production[i][0]);
                }
            }
        }
    }
}

void find_first(char c, int q1, int q2)
{
    int j;

    if (!(isupper(c))) {
        first[n++] = c;
    }
    for (j = 0; j < count; j++) {
        if (production[j][0] == c) {
            if (production[j][2] == '#') {
                if (production[q1][q2] == '\0')
                    first[n++] = '#';
                else if (production[q1][q2] != '\0'
                        && (q1 != 0 || q2 != 0)) {
                    find_first(production[q1][q2], q1,
                            (q2 + 1));
                }
                else
                    first[n++] = '#';
            }
```

```
            else if (!isupper(production[j][2])) {
                first[n++] = production[j][2];
            }
            else {
                find_first(production[j][2], j, 3);
            }
        }
    }
}

void grammarfollow(char c, int c1, int c2)
{
    int k;

    if (!(isupper(c)))
        f[m++] = c;
    else {
        int i = 0, j = 1;
        for (i = 0; i < count; i++) {
            if (final_first[i][0] == c)
                break;
        }

        while (final_first[i][j] != '!') {
            if (final_first[i][j] != '#') {
                f[m++] = final_first[i][j];
            }
            else {
                if (production[c1][c2] == '\0') {
                    follow(production[c1][0]);
                }
                else {
                    grammarfollow(production[c1][c2], c1,
                            c2 + 1);
                }
            }
            j++;
        }
    }
}
```