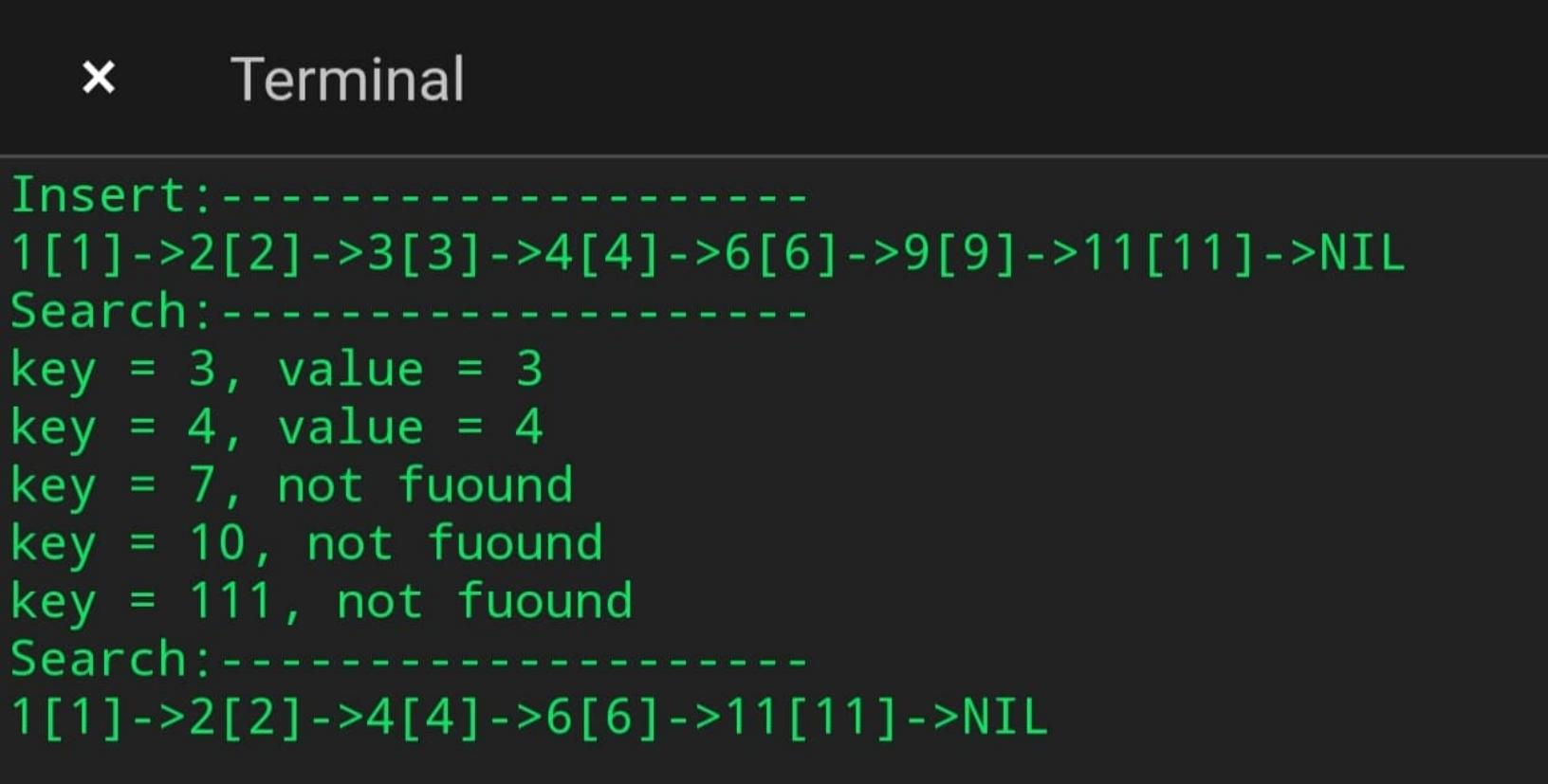
```c
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>

#define SKIPLIST_MAX_LEVEL 6

typedef struct snode {
    int key;
    int value;
    struct snode **forward;
} snode;

typedef struct skiplist {
    int level;
    int size;
    struct snode *header;
} skiplist;

skiplist *skiplist_init(skiplist *list) {
    int i;
    snode *header = (snode *) malloc(sizeof(struct snode));
    list->header = header;
    header->key = INT_MAX;
    header->forward = (snode **) malloc(
            sizeof(snode*) * (SKIPLIST_MAX_LEVEL + 1));
    for (i = 0; i <= SKIPLIST_MAX_LEVEL; i++) {
        header->forward[i] = list->header;
    }

    list->level = 1;
    list->size = 0;

    return list;
}

static int rand_level() {
    int level = 1;
    while (rand() < RAND_MAX / 2 && level < SKIPLIST_MAX_LEVEL)
        level++;
    return level;
}

int skiplist_insert(skiplist *list, int key, int value) {
    snode *update[SKIPLIST_MAX_LEVEL + 1];
    snode *x = list->header;
    int i, level;
    for (i = list->level; i >= 1; i--) {
        while (x->forward[i]->key < key)
            x = x->forward[i];
        update[i] = x;
    }
    x = x->forward[1];

    if (key == x->key) {
        x->value = value;
        return 0;
    } else {
        level = rand_level();
        if (level > list->level) {
            for (i = list->level + 1; i <= level; i++) {
                update[i] = list->header;
            }
            list->level = level;
        }

        x = (snode *) malloc(sizeof(snode));
        x->key = key;
        x->value = value;
        x->forward = (snode **) malloc(sizeof(snode*) * (level + 1));
        for (i = 1; i <= level; i++) {
            x->forward[i] = update[i]->forward[i];
            update[i]->forward[i] = x;
        }
    }
    return 0;
}

snode *skiplist_search(skiplist *list, int key) {
    snode *x = list->header;
    int i;
    for (i = list->level; i >= 1; i--) {
        while (x->forward[i]->key < key)
            x = x->forward[i];
    }
    if (x->forward[1]->key == key) {
        return x->forward[1];
    } else {
        return NULL;
    }
    return NULL;
}

static void skiplist_node_free(snode *x) {
    if (x) {
        free(x->forward);
        free(x);
    }
}

int skiplist_delete(skiplist *list, int key) {
    int i;
    snode *update[SKIPLIST_MAX_LEVEL + 1];
    snode *x = list->header;
    for (i = list->level; i >= 1; i--) {
        while (x->forward[i]->key < key)
            x = x->forward[i];
        update[i] = x;
    }

    x = x->forward[1];
    if (x->key == key) {
        for (i = 1; i <= list->level; i++) {
            if (update[i]->forward[i] != x)
                break;
            update[i]->forward[i] = x->forward[i];
        }
        skiplist_node_free(x);

        while (list->level > 1 && list->header->forward[list->level]
                == list->header)
            list->level--;
        return 0;
    }
    return 1;
}

static void skiplist_dump(skiplist *list) {
    snode *x = list->header;
    while (x && x->forward[1] != list->header) {
        printf("%d[%d]->", x->forward[1]->key, x->forward[1]->value);
        x = x->forward[1];
    }
    printf("NIL\n");
}

int main() {
    int arr[] = { 3, 6, 9, 2, 11, 1, 4 }, i;
    skiplist list;
    skiplist_init(&list);

    printf("Insert:--------------------\n");
    for (i = 0; i < sizeof(arr) / sizeof(arr[0]); i++) {
        skiplist_insert(&list, arr[i], arr[i]);
    }
    skiplist_dump(&list);

    printf("Search:--------------------\n");
    int keys[] = { 3, 4, 7, 10, 111 };

    for (i = 0; i < sizeof(keys) / sizeof(keys[0]); i++) {
        snode *x = skiplist_search(&list, keys[i]);
        if (x) {
            printf("key = %d, value = %d\n", keys[i], x->value);
        } else {
            printf("key = %d, not fuound\n", keys[i]);
        }
    }

    printf("Search:--------------------\n");
    skiplist_delete(&list, 3);
    skiplist_delete(&list, 0);
    skiplist_dump(&list);

    return 0;
}
```


Terminal
Insert:----------
1[1]->2[2]->3[3]->4[4]->6[6]->9[9]->11[11]->NIL
Search:----------
key = 3, value = 3
key = 4, value = 4
key = 7, not fuound
key = 10, not fuound
key = 111, not fuound
Search:----------
1[1]->2[2]->4[4]->6[6]->11[11]->NIL