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

#define P 999149

typedef struct list_t {
    int key;
    struct list_t * next;
} list;

list * newList();
void listInsert(list ** l, int k);
int listSearch(list * l, int val);
int listLength(list* l);
void destroyList(list* l);

typedef struct table_t {
    list ** bins;
    int size;
    int a, b;
} hashTable;

int hash(int x, int a, int b, int size);
hashTable * newHashTable(int size, int a, int b);
void hashInsert(hashTable *t, int val);
void destroyHash(hashTable *t);

list * newList() {
    return NULL;
}

void listInsert(list ** l, int k) {
    list * retVal = (list*)malloc(sizeof(list));

    retVal->key = k;
    retVal->next = *l;

    *l = retVal;
}

int listSearch(list * l, int val) {
    list *cur = l;

    for (; cur != NULL; cur = cur->next)
        if (cur->key == val)
            return 1;

    return 0;
}

int listLength(list* l) {
    if (l == NULL)
        return 0;
    return 1 + listLength(l->next);
}

void destroyList(list* l) {
    if (!l) // l == NULL
        return;

    list * succ = l->next;
    free(l);

    destroyList(succ);
}

// ------

int hash(int x, int a, int b, int size) {
    return ((a*x) + b % P) % size;
}

hashTable * newHashTable(int size, int a, int b) {
    hashTable *retVal = (hashTable*)malloc(sizeof(hashTable));

    retVal->size = size;
    retVal->a = a;
    retVal->b = b;
    retVal->bins = (list**)malloc(sizeof(list*) * size);

    int i;
    for (i = 0; i < size; i++)
        retVal->bins[i] = newList();

    return retVal;
}

void hashInsert(hashTable *t, int val) {
    int index = hash(val, t->a, t->b, t->size);

    if (listSearch(t->bins[index], val) == 0) // Non presente
        listInsert(&(t->bins[index]), val);
}

void destroyHash(hashTable *t) {
    int i;
    for (i = 0; i < t->size; i++)
        destroyList(t->bins[i]);

    free(t->bins);
    free(t);
}

void hashStats(hashTable * t) {
    int conflitti=0, maxlen = 0, numel= 0;

    int i;
    for (i = 0; i< t->size; i++) {
        int len = listLength(t->bins[i]);

        if (len > maxlen)
            maxlen = len;

        numel += len;

        if (len > 1)
            conflitti += len -1;
    }

    printf("%d\n%d\n%d\n", conflitti, maxlen, numel);
}

int main() {
    // (rand() % 9999) + 1

    int n, a, b;
    scanf("%d", &n);
    scanf("%d", &a);
    scanf("%d", &b);

    hashTable * table = newHashTable(2*n, a, b);

    int i, val;
    for(i = 0; i < n; i++) {
        scanf("%d", &val);
        hashInsert(table, val);
    }

    hashStats(table);
    destroyHash(table);
}
```