```
// Scan.c ... scanning operations

#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include "Globals.h"
#include "Scan.h"
#include "Page.h"

// startScan: start a scan on an open file
// - returns a pointer to a newly created Scan object
// - if Scan object can't be created, return NULL
Scan *startScan(int file)
{
        Scan *new = malloc(sizeof(Scan));
        if (new == NULL) return NULL;
        new->file = file;
        new->cur_page = -1; // page counter ++'d before reading
        new->cur_tup = -1; // tup counter ++'d before reading
        new->page.ntuples = -2; // less than cur_tup
        return new;
}

// nextTuple: get the tuple during a scan
// - if no more tuples, return NONE
// - if finds an invalid tuple offset, return NONE
// - otherwise copy tuple string into buf and return OK
int nextTuple(Scan *s, char *buf)
{
        if (s->cur_tup > s->page.ntuples-2) {
                do {
                        s->cur_page++;
                        if (readPage(s->file, s->cur_page, &(s-
>page)) == NONE) return NONE;
                } while (s->page.ntuples == 0);
                s->cur_tup = -1;
        }
        // fetch current tuple from current page
        s->cur_tup++;
        int tupID = s->cur_tup;
        Page *pg = &(s->page);
        int offset = pg->offset[tupID];
        // offset looks buggy
        if (offset < 0 || offset > TUPLE_BYTES_PER_PAGE-3)
                return NONE;
        strcpy(buf, &(pg->tuples[offset]));
        return OK;
}

// closeScan: clean up after a scan finishes
```

```
void closeScan(Scan *s)
{
	if (s != NULL) free(s);
}
```