CS61C F20 Quest Solutions

Instructors: Dan Garcia, Borivje Nikolic

Head TAs: Stephan Kaminsky, Cece McMahon

Q2: Bit Manipulation

Solution Walkthrough: video
bitmanip.c

```
#include <inttypes.h>
#include <stdlib.h>

//note: replace w, x, y, and z with the respective values
int GROUP_SIZE = w;
char* ROT_DIR = "right";
int ROT_AMT = x;
int ON_BIT = y;
int OFF_BIT = z;

unsigned get_bit(uint64_t x, uint64_t n) {
    return 1 & (x >> n);
}

void set_bit(uint64_t *x, uint64_t n, uint64_t v) {
    *x = (*x & ~(1 << n)) | (v << n);
}

uint64_t bit_manip(uint64_t num) {
    uint32_t group_num = 64 / GROUP_SIZE + 1;
    uint64_t mask = 0;
    uint64_t vals[group_num];
    for (int i = 0; i < GROUP_SIZE; i++) {
        mask = mask | (1 << i);
    }

    for (int i = group_num; i > 0; i--) {
        vals[i - 1] = num & mask;
        num = num >> GROUP_SIZE;
    }

    uint64_t total = 0;

    for (int i = 0; i < group_num; i++) {
        total = total << GROUP_SIZE;
        for (int j = 0; j < ROT_AMT; j++) {
            unsigned zero_bit = get_bit(vals[i], 0);
            vals[i] = vals[i] >> 1;
            set_bit(&(vals[i]), GROUP_SIZE - 1, zero_bit);
        }

        set_bit(&(vals[i]), ON_BIT, 1);
        set_bit(&(vals[i]), OFF_BIT, 0);

        total += vals[i];
    }
    return total;
}
```

Q3: Split

Solution Walkthrough: video
split.c

```
#include "split.h"
#include <string.h>

void *CS61C_malloc(size_t size);
void CS61C_free(void *ptr);

/*
For reference, this is the Node struct defined in split.h:
typedef struct node {
  char *data;
  struct node *next;
} Node;
*/

//keep in mind that there were different versions; the headers are all formatted in the same way
//replace all instances of arguments you didn't have with what you did
void split(Node *words, Node **consonants, Node **vowels) {
    if (!words || !consonants || !vowels) {
        return;
    }

    Node *const_head = NULL;
    Node *vowel_head = NULL;

    while(words) {
        Node* item = (Node*) CS61C_malloc(sizeof(Node));
        item->data = (char*) CS61C_malloc(sizeof(char) * (strlen(words->data) + 1));
        item->next = NULL;
        strcpy(item->data, words->data);

        if (strlen(words->data) % 2) {
            if (!vowel_head) {
                *vowels = item;
            } else {
                vowel_head->next = item;
            }
            vowel_head = item;
        } else {
            if (!const_head) {
                *consonants = item;
            } else {
                const_head->next = item;
            }
            const_head = item;
        }

        Node *temp = words->next;

        CS61C_free(words->data);
        CS61C_free(words);
        words = temp;
    }
    return;
}
```