Lab2

Exercise 0: Makefiles

由`Lab1`的体验，我们可以感受到在终端中编译程序是很麻烦地。 因为有很多`dependencies`的缘故，我们不得不一行一行地敲入 以确保正确的编译顺序。

`Makefile`可以解决我们的困难， 在`Makefile`中我们写下我们我们想要的编译规则，然后简单地敲入`make`即可。

Exercise 1: Bit Operations

只能用`bitwise operations`

```
#include <stdio.h>
#include "bit_ops.h"

// Return the nth bit of x.
// Assume 0 <= n <= 31
unsigned get_bit(unsigned x,unsigned n) {

    return (x>>n)&1;
}
// Set the nth bit of the value of x to v.
// Assume 0 <= n <= 31, and v is 0 or 1
void set_bit(unsigned * x,
             unsigned n,
             unsigned v) {
    unsigned mask  =  ~(1<<n);
    *x = *x &mask;
    unsigned a  = v<<n;
    *x = *x|a;
}
// Flip the nth bit of the value of x.
// Assume 0 <= n <= 31
void flip_bit(unsigned * x,
              unsigned n) {
    unsigned mask  = ~(1<<n);
    *x = *x &mask;
    unsigned a = 1<<n;
    *x = *x&a;
}

```

Exercise 2: Linear Feedback Shift Register

那原数右移动一位，然后把最右边那位弄成 之前在位置`0,2,3,5`上的数的异或和.

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

```c
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include "lfsr.h"
unsigned get_bit(unsigned x,unsigned n) {

    return (x>>n)&1;
}

void lfsr_calculate(uint16_t *reg) {
    /* YOUR CODE HERE */
    unsigned int shiftInNum = get_bit(*reg,0) ^ get_bit(*reg,2) ^ get_bit(*reg,3) ^ get_bit(*reg,5);
    *reg >>= 1;
    *reg |= shiftInNum << 15;
}
```

Exercise 3: Linked Lists

我们需要实现两个函数

- `append_node()`：在链表屁股后面添加一个新的节点。比较简单
- `reverse_list()`： 把链表翻转，不能搞一个新的链表。拿两个指针，`pre,cur`，其中`pre` 紧随`cur` 其后， 把`cur->next` 赋为`pre` 即可。

```c
#include "list.h"

/* Add a node to the end of the linked list. Assume head_ptr is non-null. */
void append_node (node** head_ptr, int new_data) {
    /* First lets allocate memory for the new node and initialize its attributes */
    /* YOUR CODE HERE */
    node *new_node = (node *)malloc(sizeof(node));
    new_node->val = new_data;
    new_node->next = NULL;

    /* If the list is empty, set the new node to be the head and return */
    if (*head_ptr == NULL) {
         *head_ptr = new_node;
        return;
    }
    node* curr = *head_ptr;
    while (curr->next != NULL) {
        curr = curr->next;
    }
    curr->next = new_node;
}

/* Reverse a linked list in place (in other words, without creating a new list).
   Assume that head_ptr is non-null. */
```

```c
void reverse_list (node** head_ptr) {
    node* prev = NULL;
    node* curr = *head_ptr;
    node* next = NULL;
    while (curr != NULL) {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    /* Set the new head to be what originally was the last node in the list */
    *head_ptr = prev;
}
```

Exercise 4: Memory Management

- `bad_vector_new()` 错在哪里？没有给 `v` 分配 `heap` 上的空间，而且给 `data` 分配空间的格式错了。
- `also_bad_vector_new()` 错在哪里？没有给 `v` 分配 `heap` 上的空间，而且给 `data` 分配空间的格式错了。

最后我们要修改 `vector.h` 和 `makefile`，由于我还不会 `makefile` 暂时先跳过。