实验一 可变分区存储管理

一、实验题目

编写一个C语言程序，模拟UNIX的可变分区内存管理，使用**循环首次适应法**实现对一块内存区域的分配和释放管理。

二、实验目的

1. 加深对可变分区存储管理的理解。
2. 考察使用C语言编写代码的能力，特别是C语言编程的难点之一：指针的使用。
3. 复习使用指针实现链表以及在链表上的基本操作。

三、实验要求

1、一次性向系统申请一块大内存空间 (e.g. 1KB)

使用$malloc()$函数进行内存申请，作为之后管理的内存空间。

2、使用合适的数据结构实现空闲存储区

- 结构数组（**基础**）

```
struct map{
    unsigned m_size;
    char *m_addr;
};
```

- 双向链表（**进阶**）

```
struct map{
    unsigned m_size;
    char *m_addr;
    struct map *next, *prior;
};
```

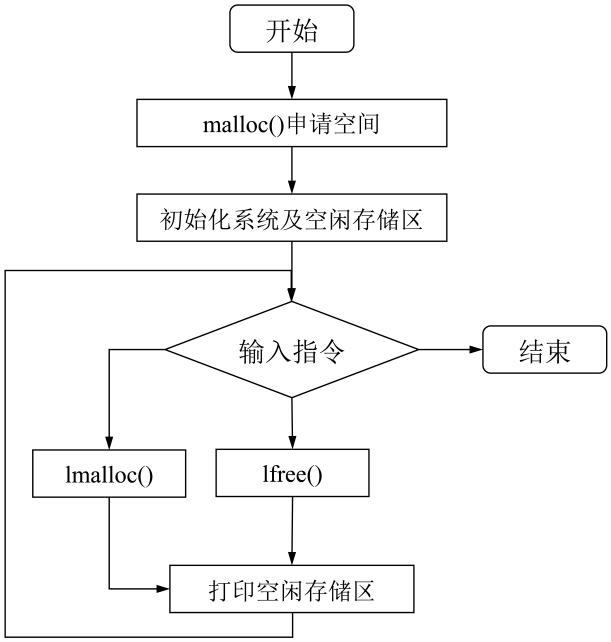
3、主要实现如下两个内存分配函数

```
addr = (char *)lmalloc(unsigned size);
lfree(unsigned size, char *addr);
```

其中，参数$size$以及$addr$均通过控制台窗口输入，还可以传入其他需要的参数。

执行完一次内存分配或释放后，需要将当前的空闲存储区的情况打印出来。

4、整体框架



四、注意事项

1. 通过键盘输入的指令**可以**是如下形式，或其他类似形式：

   `m[alloc] 100`。表示通过$lmalloc()$函数申请得到100字节的内存空间。

   `f[ree] 100 2567899`。表示通过$lfree()$函数，将起始地址为2567899的内存释放掉，大小为100字节。$lfree()$也可以考虑使用**相对地址**进行管理。

2. 读取输入命令可以使用：

   ```
   scanf("%c", &cmdchar);
   ```

   ```
   cmdchar = getchar();
   ```

   读取$size$和$addr$时，可以使用：

   ```
   scanf("%u", &size);
   ```

   ```
   scanf("%u %u", &size, &addr);
   ```

3. 在读取输入指令时注意滤去输入缓冲区内的'\n'，'\t'等空白字符：

   ```
   do
       c=getchar();
   while(c == '\n' || c == '\t' || c == ' ');
   ```

五、程序测试

可以在Linux或Windows环境中进行测试：

- [ ] 保证$lmalloc()$满足“循环”、”首次”的特征。
- [ ] $lfree()$应考虑和临近空闲分区的四种情况。
- [ ] 一些可能的边界情况及出错处理。

还可以手动设计测试文件，使用**I/O重定向**将之输入程序，并将输出结果写入另一文件：

```
myproc < input > result
```

六、实验报告要求

最终提交的实验报告至少包括：

1. 题目
2. 算法思想及概要设计
3. 重要模块的功能、详细设计以及接口说明
4. 重要数据结构及变量的说明
5. 测试方法及结果
6. 结果及错误的分析

七、提交要求

4月11日23:59之前将**实验报告**和**源代码**（需包含必要的注释）打包提交至CANVAS。