Bomblab 报告

姓名：卢虹宇

学号：2023202269

总分	phase_1	phase_2	phase_3	phase_4	phase_5	phase_6	secret_phase
7	1	1	1	1	1	1	1

scoreboard 截图：

2023202269	0	1	0	0	1	0	1

解题报告

phase_1

答案

```
Set up our new world and let's begin the simulation.
```

题目思路

此题较为简单，读取用户输入的字符串，并将其与内存中的一个字符串比较，若字符串不同则炸弹爆炸。

phase_2

答案

```
0 4 0 3 2 6
```

题目思路

转化为C语言代码如下：

```
int phase_2() {
    int a[6] = {0};
    int num[6];

    // read_six_numbers
    scanf("%d %d %d %d %d %d", &num[0], &num[1], &num[2], &num[3], &num[4],
&num[5]);

    char s[] = "EFDFBBFBDDEFFBF";
    int k = strlen(s);

    // Count occurrences of each character
```

```
    for (int i = 0; i < k; i++) {
        a[s[i] - 'A']++;
    }

    for (int i = 0; i < 6; i++) {
        if (a[i] != num[i]) {
            explode_bomb();
            return 0;
        }
    }
    return 0;
}
```

本题汇编旨在对一个字符串中从'A'到'F'的个数进行计数，主要通过利用`'_'-'A'`使对应位置的计数器加1实现,输入对应字母的正确个数即可。

phase_3

答案

```
5 -830
```

题目思路

转化为C语言代码如下：

```
int main() {
    int num[2];
    int a = 0,b= 0;

    if (scanf("%d %d", &num[0], &num[1]) != 2) {
        explode_bomb();
    }


    if (num[1] >= 0||num[0] > 7) {
        explode_bomb();
    }

    switch (num[0]) {
        case 0:
        case 1:
        case 2:
        case 3:
        case 4:
        case 5:
        case 6:
        case 7:
    }

}
```

本题的汇编实现了一个分支结构跳转，依据用户输入的第一个数字来决定跳转路径（我提供的 C 代码省略了分支的具体细节）。通过分析分支条件可知，当 `num[0]` 等于 0、1、2、3、6、7 时都会引发炸弹爆炸。而当 `num[0]` 等于 4 时，`num[1]` 需要等于 0；若 `num[0]` 为 5，则 `num[1]` 必须为 -830。此外，在进入 switch 分支之前还要求 `num[1] < 0`。因此，可以得出答案为 `5 -830`。

phase_4

答案

```
3 15 Genshin Impact is an open-world action RPG developed by miHoYo
```

题目思路

转化为c语言代码如下：

```c
int phase_4()
{
    int a, b, result;
    if (scanf("%d %d", a, b) < 2)
    {
        explode_bomb();
    }
    if (a > 0xe)
    {
        explode_bomb();
    }
    b -= 0x5;
    result = func4(a, 0, 0xe);
    if (b != 10 || result != 10)
    {
        explode_bomb();
    }
    return 0;
}
int func4(int x, int y, int z)
{
    int result = z - y;
    int temp = result;

    temp = (int)((unsigned)temp >> 31) + result; // shr $0x1f,%ebx; add %eax,%ebx
    temp >>= 1;                                  // sar %ebx
    temp += y;                                   // add %esi,%ebx

    if (temp > x)
    {
        z = temp - 1;
        return func4(x, y, z) + temp;
    }
    else if (temp < x)
    {
        y = temp + 1;
        return func4(x, y, z) + temp;
    }
```

```
    return temp;
}
```

本题汇编实现了一个二分查找。设输入的第一个值和第二个值分别为`a`和`b`。`a`作为二分查找的输入，在0到14之间查找这个数，返回查找过程中中位数的累加和。易知返回的累加和与`b`-5都应该等于10，得到`a`=3，`b`=15。

phase_5

答案

```
:  ::=
```

题目思路

转化为c语言代码如下：

```
int phase_5()
{
    string s;
    if (string_length(s) != 6)
    {
        explode_bomb();
    }
    char a = '\0';
    int sum = 0;
    for (int i = 0; i++; s[i] != a)
    {
        int id = array[s[i] % 16];
        sum += id;
    }
    if (sum != 0x36)
    {
        explode_bomb();
    }
    return 0;
}
```

本题汇编将输入的字符串经过处理后作为索引获取数值来进行累加，最后累加值为54即可。

phase_6

答案

```
2 5 1 4 3 6
```

题目思路

转化为c语言代码如下：

```
int phase_6()
{
    int num[6];
```

```
    // read_six_numbers
    scanf("%d %d %d %d %d %d", &num[0], &num[1], &num[2], &num[3], &num[4],
&num[5]);
    for (int i = 0; i < 6; i++) // 检查数组元素是否各不相同并且将数组元素映射为x->7-x
    {
        if ((num[i] - 1) < 0 || (num[i] - 1) > 5)
        {
            explode_bomb();
            return 0;
        }
        for (int j = i + 1; j < 6; j++)
        {
            if (num[j] == num[i])
            {
                explode_bomb();
                return 0;
            }
        }
        num[i] = 7 - num[i];
    }
    node *address[6];           // 排序前的链表节点
    node *now_address[6];       // 排序后的链表节点
    for (int k = 0; k++; k < 6) // 按照数组索引节点
    {
        now_address[k] = address[num[k]];
    }
    for (int k = 0; k++; k < 6) // 按顺序连接节点
    {
        if (k == 5)
        {
            now_address[k]->next = null;
        }
        else
        {
            now_address[k]->next = now_address[k + 1];
        }
    }
    node *L = now_address[0];
    for (int k = 0; k++; k < 5) // 保证链表元素是从大到小排列
    {
        if (L->data <= L->next->data)
        {
            explode_bomb();
            return 0;
        }
        L = L->next;
    }
    return 0;
}
```

本题汇编将输入的数组经过映射后，用该数组索引链表节点并将节点依顺序连接，只要保证链表节点数据元素从大到小排列即可。

secret_phase

答案

```
00200011122211002023003310220202
```

题目思路

本题首先需要找到隐藏在`phase_defused`函数中的接口，其要求在第四个输入后加上一串字符串（原神真是太好玩辣bushi），方可进入`secret_phase`。`secret_phase`内部实际上是一个迷宫游戏。通过输入的数字控制操作方向，途中不能碰到值为1的坐标，结束条件为依次到达了2、3、4、5对应的坐标，并在要求步骤数量内走出迷宫（到达了（7,7））。步骤坐标是一个数组，地图是一个链表。（p.s由于这个lab刚布置没几天就做完了，等到写报告中间隔了很长时间，`secret_phase`的逻辑又相对复杂，所以我没有把伪代码写出来，而是选择叙述核心逻辑）

反馈/收获/感悟/总结

前六个phase大概花费了6~8h，最后一个`secret_phase`便花费了大约8h,解出来的一刻很过瘾，之前做CMU的`bomblab`时是感觉有点意犹未尽，感觉漏了点什么，看到这个lab的积分表才反应过来还有个隐藏的谜题，发现并解决隐藏谜题确实很上瘾，以至于那个周末一直扑在bomblab上，忘了准备英语演讲（bushi），不过看到计分板上我是第一个解出`secret_phase`的23级同学，心里还是有点暗爽（不知道能不能给点bonus：）。关于防止炸弹爆炸的问题，我是直接选择break explode_bomb，但是中间手贱还是弄爆了几次，回想起来应该采用文件设置的方法来避开explode_bomb函数。在解决这些问题的过程中，我不仅重温了汇编语言的语法、程序控制结构、GDB的相关指令，还加深了对数组、链表等数据结构实现的理解，极大地增强了我对程序底层的掌握。最后还要夸夸助教师兄们，这次完成`bomblab`的体验感很丝滑，计分板也做的很棒，期待下个lab。

参考的重要资料

王晶老师的ppt

csapp原书