1 Phase One

Answer: *"Border relations with Canada have never been better."*

```
1  void phase_1(char *input)
2  {
3      char* phase_1_secret = "Border relations with Canada have never been better.";
4
5      if (strings_not_equal(input, phase_1_secret))
6      {
7          explode_bomb();
8      }
9  }
```

Listing 1: Phase 1

2 Phase Two

Answer: *"1 2 4 8 16 32"*

```
1  void phase_2(char *input)
2  {
3      int numbers[6];
4      int numberCount = read_six_numbers(input, numbers);
5      if (numberCount < 6)
6      {
7          explode_bomb();
8      }
9
10     if (numbers[0] != 1)
11     {
12         explode_bomb();
13     }
14
15     int i = 1;
16     while (i < 6)
17     {
18         if (numbers[i] != 2 * numbers[i - 1])
19         {
20             explode_bomb();
21         }
22     }
23 }
```

Listing 2: Phase 2

3 Phase Three

Answer: Any string with following **a** and **k** combinations, for example:
"3 256"

```
1  void phase_3(char *str)
2  {
3      int a, b;
4      char *format ="%d %d";
5      int numbersCount = sscanf(str, format, &a, &b);
6      if (numbersCount < 2)
7      {
8          explode_bomb();
9      }
10
11     if ((unsigned int)a > 7)
12     {
13         explode_bomb();
14     }
15
16     int k = 0;
17     switch (a)
18     {
19         case 0:
```

```
20              // k = 207
21              k = 0xcf;
22              break;
23          case 1:
24              // k = 311
25              k = 0x137;
26              break;
27          case 2:
28              // k = 707
29              k = 0x2c3;
30              break;
31          case 3:
32              // k = 256
33              k = 0x100;
34              break;
35          case 4:
36              // k = 389
37              k = 0x185;
38              break;
39          case 5:
40              // k = 206
41              k = 0xce;
42              break;
43          case 6:
44              // k = 767
45              k = 0x2ff;
46              break;
47          default:
48              explode_bomb();
49              break;
50      }
51
52      if (k != b)
53      {
54          explode_bomb();
55      }
56  }
```

Listing 3: Phase 3

4 Phase Four

Answer: *"7 0"*

```
1   void phase_4(char *str)
2   {
3       int a, b;
4       char *format = "%d %d";
5       int numbersCount = sscanf(str, format, &a, &b);
6       if (numbersCount != 2)
7       {
8           explode_bomb();
9       }
10
11      if ((unsigned int)a > 0xe)
12      {
13          explode_bombe();
14      }
15
16      int k = func4(a, 0, 0xe);
17      if (k != 0)
18      {
19         explode_bomb();
20      }
21
22      if (b != 0)
23      {
24          explode_bomb();
25      }
26  }
```

Listing 4: Phase 4

func4 It is an additional function intended for the obfuscation of function *phase4()* logic.

```
int func4(int num, int a, int b)
{
    int dif = b - a;
    int k = dif + ((unsigned int)dif >> 31);
    k = k >> 1;
    k = k + 1 * a;
    if (k > num)
    {
        return 2 * func4(num, a, k - 1);
    }

    if (k >= num)
    {
        return 0;
    }

    return 1 + 2 * func4(num, k + 1, k);
}
```

Listing 5: func4

5 Phase Five

Answer: *"IONEFG"*

```
void phase_5(char *str)
{
    if (string_length(str) != 6)
    {
        explode_bomb();
    }

    int i = 0;
    char builtString[7];
    char *template = "maduiersnfotbylSo you think you can stop the bomb with
        ctrl+c, do you?";
    while (i != 6)
    {
        int offset = str[i] & 0xff;
        builtString[i] = template[offset];
    }

    builtString[7] = '\0';
    char *original = "flyers";
    if (strings_not_equal(builtString, original))
    {
        explode_bomb();
    }
}
```

Listing 6: Phase 5

6 Phase Six

Answer: *"4 3 2 1 6 5"*

```
void phase_6(char *str)
{
    int numbers[6];
    read_six_numbers(str, numbers);
    int i;
    for (i = 0; i < 6; i++)
    {
        if ((unsigned int)numbers[i] - 1 > 5)
        {
            explode_bomb();
        }

```

```
13          int j;
14          for (j = i; j < 6; j++)
15          {
16              if (numbers[j] == numbers[i])
17              {
18                  explode_bomb();
19              }
20          }
21      }
22
23      for (i = 0; i < 6; i++)
24      {
25          numbers[i] = 7 - numbers[i];
26      }
27
28      int **memory[6];
29      int v6 = 0x1bb;
30      int v5 = 0x1dd;
31      int v4 = 0x2b3;
32      int v3 = 0x39c;
33      int v2 = 0xa8;
34      int v1 = 0x142;
35
36      void **p6A[] = {v6, 0x0};
37      void **p5A[] = {v5, p6A};
38      void **p4A[] = {v4, p5A};
39      void **p3A[] = {v3, p4A};
40      void **p2A[] = {v2, p3A};
41      void **p1A[] = {v1, p2A};
42
43      for (i = 0; i < 6; i = i++)
44      {
45          int number = numbers[i];
46          void **current = p1A;
47          while (number > 1)
48          {
49              current = *(current + 1);
50              number--;
51          }
52
53          memory[i] = current;
54      }
55
56      for (i = 0; i < 5; i++)
57      {
58          *(memory[i] + 1) = memory[i + 1];
59      }
60
61      for (i = 0; i < 5; i++)
62      {
63          if ((int)*memory[i] <= (int)(*(*(memory[i] + 1))))
64          {
65              explode_bomb();
66          }
67      }
68  }
```

Listing 7: Phase 6

7 Utility Functions

7.1 String Length

```
1  int string_length(char *str)
2  {
3      int length = 0;
4      char *current = str;
5      while (*current == '\0')
6      {
7          current++;
8          length = current - str;
9      }
10
11     return length;
12 }
```

Listing 8: String Length

7.2 Strings Not Equal

```
1  int strings_not_equal(char *str1, char *str2)
2  {
3      int str1Length = string_length(str1);
4      int str2Length = string_length(str2);
5      if (str1Length != str2Length)
6      {
7          return 1;
8      }
9
10     char *current1 = str1;
11     char *current2 = str2;
12     while (*current1 != '\0')
13     {
14         if (*current1 != *current2)
15         {
16             return 1;
17         }
18
19         current1++;
20         current2++;
21     }
22
23     return 0;
24 }
```

Listing 9: Strings Not Equal

7.3 Read six numbers

```
1  void read_six_numbers(char *str, int numbers[])
2  {
3      char *format = "%d %d %d %d %d %d";
4      int numbersCount = sscanf(str, format,
5          &numbers[0],
6          &numbers[1],
7          &numbers[2],
8          &numbers[3],
9          &numbers[4],
10         &numbers[5]);
11
12     if (numbersCount < 6)
13     {
14         explode_bomb();
15     }
16 }
```

Listing 10: Read Six Numbers