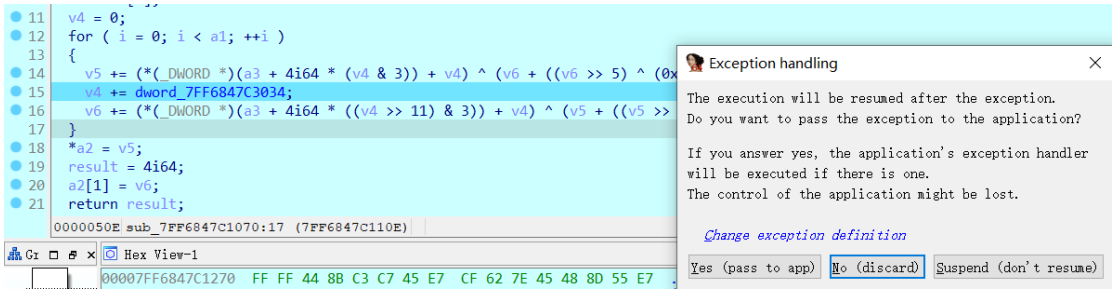
Creakme2

Ida 中找到程序伪代码

```
  sub_7FF6847C1008("%32s");
  sub_7FF6847C1070(32i64, Buf1, v11);
  sub_7FF6847C1070(32i64, v8, v11);
  sub_7FF6847C1070(32i64, v9, v11);
  sub_7FF6847C1070(32i64, v10, v11);
  Buf2[0] = 0x457E62CF;
  Buf2[1] = 0x9537896C;
  Buf2[2] = 0x1F7E7F72;
  Buf2[3] = 0xF7A073D8;
  Buf2[4] = 0x8E996868;
  Buf2[5] = 0x40AFAF99;
  Buf2[6] = 0xF990E34;
  Buf2[7] = 0x196F4086;
  v3 = memcmp(Buf1, Buf2, 0x20ui64);
  v4 = "Congratulations!";
  if ( v3 )
    v4 = "Try again!";
  puts(v4);
  return 0;
}
```

步入加密函数

```
  v5 = *a2;
  v6 = a2[1];
  v4 = 0;
  for ( i = 0; i < a1; ++i )
  {
    v5 += (*(a3 + 4i64 * (v4 & 3)) + v4) ^ (v6 + ((v6 >> 5) ^ (16 * v6)));
    v4 += dword_7FF6847C3034;
    v6 += (*(a3 + 4i64 * ((v4 >> 11) & 3)) + v4) ^ (v5 + ((v5 >> 5) ^ (16 * v5)));
  }
  *a2 = v5;
  result = 4i64;
  a2[1] = v6;
  return result;
}
```



执行第 15 行语句出现除零之后的 exception handling

```
;   __try { // __except at try_except1
;     __try { // __except at try_except2
mov     eax, cs:dword_7FF6847C3034
mov     ecx, [rsp+58h+var_38]
add     ecx, eax
mov     eax, ecx
mov     [rsp+58h+var_38], eax
mov     eax, [rsp+58h+var_38]
sar     eax, 1Fh
mov     [rsp+58h+var_28], eax
mov     eax, 1
cdq
mov     ecx, [rsp+58h+var_28]
idiv    ecx
mov     [rsp+58h+var_1C], eax
jmp     short loc_7FF6847C114E
;     } // starts at 7FF6847C1112
; ---------------------------------------------------------------------------

try_except2:                            ; DATA XREF: .rdata:00007FF6847C27C4↓o
;   __except(loc_7FF6847C1DF6) // owned by 7FF6847C1112
mov     eax, [rsp+58h+var_38]
xor     eax, 1234567h
mov     [rsp+58h+var_38], eax

loc_7FF6847C114E:                       ; CODE XREF: sub_7FF6847C1070+CF↑j
jmp     short loc_7FF6847C1158
```

```
;   } // starts at 7FF6847C1112
; ---------------------------------------------------------------------------

try_except1:                            ; DATA XREF: .rdata:00007FF6847C27D4↓o
;   __except(loc_7FF6847C1E21) // owned by 7FF6847C1112
mov     [rsp+58h+var_38], 9E3779B1h

loc_7FF6847C1158:                       ; CODE XREF: sub_7FF6847C1070:loc_7FF6847C114E↑j
mov     eax, [rsp+58h+var_34]
shl     eax, 4
mov     ecx, [rsp+58h+var_34]
shr     ecx, 5
xor     eax, ecx
add     eax, [rsp+58h+var_34]
mov     ecx, [rsp+58h+var_38]
sar     ecx, 0Bh
and     ecx, 3
movsxd  rcx, ecx
mov     rdx, [rsp+58h+arg_10]
mov     ecx, [rdx+rcx*4]
mov     edx, [rsp+58h+var_38]
add     edx, ecx
mov     ecx, edx
xor     eax, ecx
mov     ecx, [rsp+58h+var_30]
add     ecx, eax
mov     eax, ecx
mov     [rsp+58h+var_30], eax
jmp     loc_7FF6847C10BE
```

以上为 try-except 结构

Idiv ecx //ecx 为 64 位

```
sar     eax, 1Fh
mov     [rsp+58h+var_28], eax
mov     eax, 1
cdq
mov     ecx, [rsp+58h+var_28]
idiv    ecx
```

如果 eax 算数右移 0x1f 后为 0，则跳转到异常处理

```
try_except2:                            ; DATA XREF: .rdata:00007FF6847C27C4↓o
;   __except(loc_7FF6847C1DF6) // owned by 7FF6847C1112
mov     eax, [rsp+58h+var_38]
xor     eax, 1234567h
mov     [rsp+58h+var_38], eax
```

编写 c 解密代码

```
    #include<stdio.h>
int main()
{
    unsigned int buf2[8] =
{ 0x457E62CF,0x9537896C,0x1F7E7F72,0xF7A073D8,0x8E996868,0x40AFAF99,0xF990E34,0x196F408
6};
    int v4[33];
    int v11[10] = {1,2,3,4,5,6,7,8,9,0};
    int i, j;
    unsigned int  v5, v6;
    int k;
    v4[0]=0,v4[1] = 0x9E3779B1;
    for (k = 2; k <= 32; k++)
    {
        v4[k] = v4[k - 1] + 0x9E3779B1;
        if (v4[k] >> 0x1f == 0)
        {
            v4[k] ^= 0x1234567;
        }
    }
    for (j = 0; j <= 3; j++)
    {
        v5 = buf2[j * 2], v6 = buf2[j * 2 + 1];
        for (i = 31; i >= 0; i--)
        {
            v6 -= (v11[(v4[i+1] >> 11) & 3] + v4[i+1]) ^ (v5 + ((v5 >> 5) ^ (16 *
v5)));
            v5 -= (v11[v4[i] & 3] + v4[i]) ^ (v6 + ((v6 >> 5) ^ (0x10 * v6)));
        }
        printf("%x%x", v5, v6);
    }
```

```
    return 0;
}
```

```
6d61676845537b6530735f4835646e755f30355f65746e31743565727d676e69
```

得到输出
转换成字符后为 maghES{e0s_H5dnu_05_etn1t5er}gni
每四位进行倒序整理后得到 flag
hgame{SEH_s0und5_50_1ntere5ting}

fakeshell
使用 ida 打开
找到 sudo 的 password 输入后校验的值

```
23  code[0] = 0xE0B25F3D8FFA94B6LL;
24  code[1] = 0xE79D6C9866D20FEALL;
25  code[2] = 0x6D6FBEC57140081BLL;
26  code[3] = 0xF6F3BDA88D097B7CLL;
27  v8 = 0;
28  for ( i = 0; i <= 31; ++i )
29  {
30    if ( *((_BYTE *)code + i) != *((_BYTE *)input + i) )
31      return 0LL;
32  }
33  return 1LL;
34 }
```

```
17  input[0] = *a1;
18  input[1] = v2;
19  v3 = a1[3];
20  input[2] = a1[2];
21  input[3] = v3;
22  sub_5630860C8635(v9, input, 32LL);
```

22 行是加密函数，对输入进行加密

步入加密函数

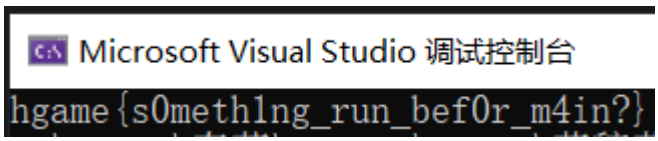
```
11  for ( i = 0LL; ; ++i )
12  {
13    result = i;
14    if ( i >= a3 )
15      break;
16    v5 = (v5 + 1) % 256;
17    v6 = (v6 + *(unsigned __int8 *)(v5 + a1)) % 256;
18    v4 = *(_BYTE *)(v5 + a1);
19    *(_BYTE *)(v5 + a1) = *(_BYTE *)(v6 + a1);
20    *(_BYTE *)(a1 + v6) = v4;
21    *(_BYTE *)(a2 + i) ^= *(_BYTE *)((unsigned __int8)(*(_BYTE *)(v5 + a1) + *(_BYTE *)(v6 + a1)) + a1);
22  }
23  return result;
24 }
```

我们发现加密语句对输入的每一位是单独加密的

故可以编写 c 解密代码如下

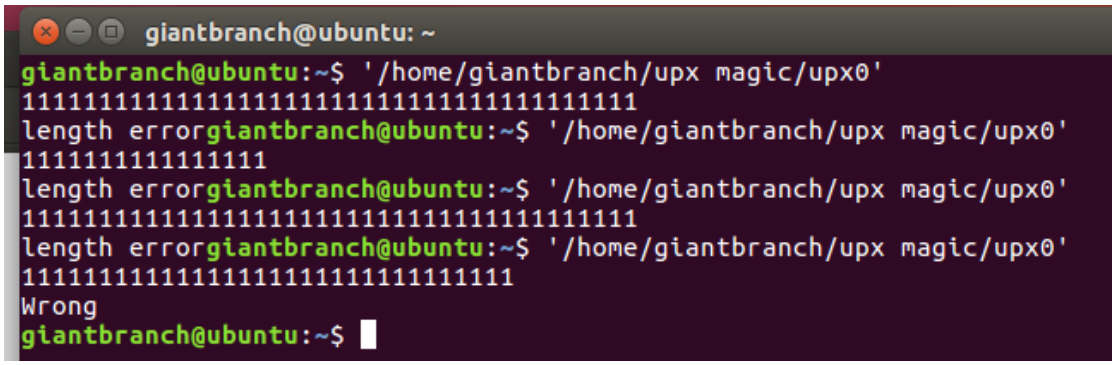
```
#include<stdio.h>
int main()
{
    int codet[300] =
{ 0x6C,0x0A8,0x54,0x0D3,0x0D1,0x0FC,0x87,0x2F,0x0F7,0x0E4,0x74,0x5F,0x1B,0x0A4,0x22,0x6A,0x0EF,0x17,0x4F,0x4,0x0B4,0x3D,0x40,0x36,0x0A0,0x32,0x5B,0x1D,0x8A,0x57,0x0AD,0x0FD,0x7D,0x0F6,0x48,0x0E2,0x7F,0x0D4,0x1A,0x1F,0x15,0x9F,0x0C0,0x89,0x0BB,0x3F,0x3A,0x73,0x28,0x0,0x1C,0x0A3,0x2E,0x6D,0x68,0x0C5,0x0E,0x18,0x90,0x0D,0x0A,0x0D6,0x4D,0x45,0x0FF,0x0DB,0x11,0x0DA,0x95,0x53,0x5A,0x72,0x2D,0x0A1,0x0F,0x50,0x7A,0x0B0,0x0BC,0x8D,0x0BA,0x0CC,0x56,0x88,0x0CF,0x0CB,0x0C7,0x26,0x80,0x42,0x9E,0x7C,0x7,0x0F0,0x0E9,0x49,0x0DF,0x71,0x98,0x6B,0x0B1,0x0B5,0x0E7,0x0F8,0x67,0x24,0x0BF,0x46,0x77,0x0E5,0x8E,0x0B,0x29,0x63,0x85,0x34,0x62,0x0D2,0x4E,0x0ED,0x0A7,0x41,0x8C,0x0D7,0x43,0x60,0x0D9,0x0B8,0x0EC,0x0D5,0x0B6,0x92,0x8,0x0C1,0x5D,0x86,0x0C,0x44,0x7B,0x0CA,0x0AB,0x0E0,0x96,0x83,0x2A,0x3C,0x0B7,0x0DD,0x0DC,0x3B,0x19,0x99,0x0D0,0x0A9,0x0DE,0x0C6,0x2,0x0D8,0x5C,0x0F3,0x52,0x9B,0x9,0x64,0x30,0x91,0x0C9,0x0C3,0x0F2,0x2C,0x25,0x0E6,0x9C,0x0EE,0x10,0x13,0x81,0x20,0x59,0x0FE,0x0FB,0x0AA,0x0CD,0x16,0x27,0x76,0x0FA,0x33,0x0B9,0x0E1,0x1E,0x0F5,0x4C,0x0EA,0x0F1,0x0BE,0x0F4,0x5,0x0A2,0x93,0x2B,0x0A5,0x12,0x0A6,0x21,0x0E8,0x51,0x0CE,0x79,0x6F,0x66,0x9D,0x84,0x1,0x5E,0x8F,0x6E,0x9A,0x3E,0x0AE,0x7E,0x6,0x14,0x0EB,0x82,0x0E3,0x97,0x69,0x35,0x23,0x61,0x0B2,0x0B3,0x94,0x3,0x39,0x0C4,0x47,0x0BD,0x0AC,0x78,0x55,0x0AF,0x37,0x0C2,0x4A,0x70,0x65,0x75,0x8B,0x31,0x0C8,0x4B,0x38,0x58,0x0F9,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0,0x0B2,0x58,0x9,0x61,0x62,0x46,0x0A2, };
    int realcode[40] =
{ 0XB6,0X94,0XFA,0X8F,0X3D,0X5F,0XB2,0XE0,0XEA,0X0F,0XD2,0X66,0X98,0X6C,0X9D,0XE7,0X1B,0X08,0X40,0X71,0XC5,0XBE,0X6F,0X6D,0X7C,0X7B,0X09,0X8D,0XA8,0XBD,0XF3,0XF6 };
    int i,v5 = 0,v6 = 0,temp,j=0,jk=0;
    for (i = 0; i < 32; i++)
    {
        v5 = (v5 + 1)%0x100;
        v6 = (codet[v5] + v6) % 0x100;
        temp = codet[v5], codet[v5] = codet[v6], codet[v6] = temp;
        for (j = 0; j <= 0xff; j++)
        {
            jk = j;
            jk ^= codet[(codet[v5] + codet[v6])&0xff];
            if (jk == realcode[i])
            {
                printf("%c",j);
                break;
            }
        }
        if (j == 0xff+1) printf("\n{%d}\n", i);
    }
    return 0;
```

}



得到 flag
hgame{s0meth1ng_run_bef0r_m4in?}

upxmagic0



输入 32 个 1 后不再出现 length error 而是 wrong
可知 flag 为 32 位

```
for ( i = 0; i < (unsigned __int64)sub_4004E0(v16); ++i )
{
  v12 = *((char *)v16 + i) << 8;
  for ( j = 0; j <= 7; ++j )
  {
    if ( (v12 & 0x8000) != 0 )
      v12 = (2 * v12) ^ 0x1021;
    else
      v12 *= 2;
  }
  v15[i] = (unsigned __int16)v12;
}
```

对输入进行加密
我们可以编写 c 语言代码对所有 ASC 字符进行上述的加密
随后与下列校验的值进行比较得到最后的 flag

```
v14[0] = 36200;
v14[1] = 40265;
v14[2] = 10770;
v14[3] = 43802;
v14[4] = 52188;
v14[5] = 47403;
v14[6] = 11826;
v14[7] = 40793;
v14[8] = 56781;
v14[9] = 40265;
v14[10] = 43274;
v14[11] = 3696;
v14[12] = 62927;
v14[13] = 2640;
v14[14] = 23285;
v14[15] = 65439;
v14[16] = 40793;
v14[17] = 48395;
v14[18] = 22757;
v14[19] = 14371;
v14[20] = 48923;
v14[21] = 30887;
v14[22] = 43802;
v14[23] = 18628;
v14[24] = 43274;
v14[25] = 11298;
v14[26] = 40793;
v14[27] = 23749;
v14[28] = 24277;
v14[29] = 30887;
v14[30] = 9842;
v14[31] = 22165;
```

```
#include<stdio.h>
int main()
{
    int i,j,k,a[128];
    int code[33] =
{ 0x8D68 ,0x9D49 ,0x2A12 ,0xAB1A ,0xCBDC ,0xB92B ,0x2E32 ,0x9F59 ,0xDDCD ,0x9D
49 ,0xA90A ,0xE70 ,0xF5CF ,0xA50 ,0x5AF5 ,0xFF9F ,0x9F59 ,0xBD0B ,0x58E5 ,0x3
823 ,0xBF1B ,0x78A7 ,0xAB1A ,0x48C4 ,0xA90A ,0x2C22 ,0x9F59 ,0x5CC5 ,0x5ED5 ,
0x78A7 ,0x2672 ,0x5695 };
    for (i = 0; i <= 127; i++)
    {
        k = i << 8;
        for (j = 0; j <= 7; ++j)
```

```
        {

            if ((k & 0x8000) != 0)
            {
                k = (2 * k) ^ 0x1021;
            }
            else
            {
                k *= 2;
            }
        }
        k = k & 0xffff;
        a[i] = k;
    }
    for (i = 0; i < 32; i++)
    {
        for (j = 0; j <= 127; j++)
        {
            if (code[i] == a[j])
            {
                printf("%c", j);
                break;
            }
        }
    }
    return 0;
}
```

编写解密代码

```
noW_YOu^koNw-UPx^mAG|C_@Nd^crC16
```

得到 flag

noW_YOu~koNw-UPx~mAG|C_@Nd~crC16

upxmagic1

通过 strings 命令看到壳为 upx3.94

使用 winhex 打开

将

```
55 50 58 3F  00 00 00 00 55 50 58 3F    UPX?    UPX?
```

改 为

```
55 50 58 21  00 00 00 00 55 50 58 21    UPX!    UPX!
```

使用 upx3.94 成功脱壳

```
C:\Users\春芹>D:\TOOLS\upx\upx394w\upx394w\upx.exe -d "C:\Users\春芹\Desktop\upx magic1\upx2"
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2017
UPX 3.94w       Markus Oberhumer, Laszlo Molnar & John Reiser   May 12th 2017

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
    885352 <-    394032   44.51%   linux/amd64   upx2

Unpacked 1 file.
```

找到加密函数

```
for ( i = 0; i < (unsigned __int64)sub_4004E0(v16); ++i )
{
  v12 = *((char *)v16 + i) << 8;
  for ( j = 0; j <= 7; ++j )
  {
    if ( (v12 & 0x8000) != 0 )
      v12 = (2 * v12) ^ 0x1021;
    else
      v12 *= 2;
  }
  v15[i] = (unsigned __int16)v12;
}
```

编写解密代码

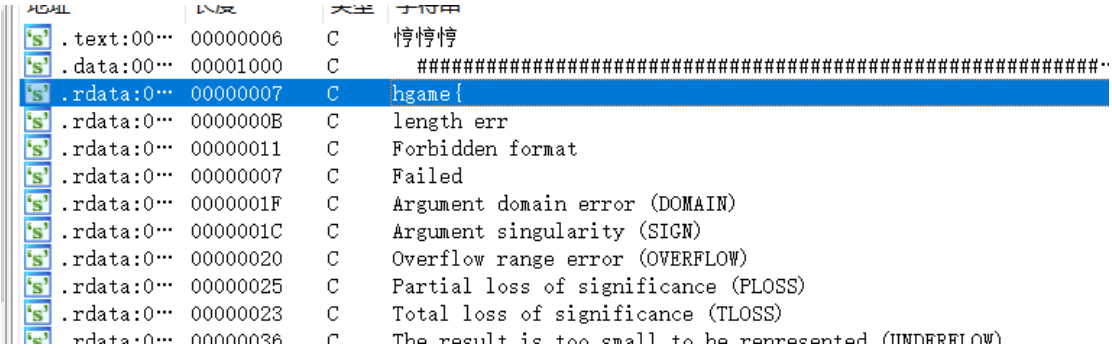
```
#include<stdio.h>
int main()
{
    int i, j, k, a[128];
    int code[33] = { 36200,  40265,  10770,  43802,  52188,  47403,  11826,  40793,
56781,  40265,  43274,  3696,  62927,  24277,  15363,  31879,  9842,  43802,  2640,
23285,  65439,  40793,  48395,  22757,  14371,  48923,  30887,  43802,  18628,  43274,
11298,  40793,  23749,  24277,  30887,  9842,  22165 };
    for (i = 0; i <= 127; i++)
    {
        k = i << 8;
        for (j = 0; j <= 7; ++j)
        {

            if ((k & 0x8000) != 0)
            {
                k = (2 * k) ^ 0x1021;
            }
            else
            {
                k *= 2;
            }
```

```
        }
        k = k & 0xffff;
        a[i] = k;
    }
    for (i = 0; i < 37; i++)
    {
        for (j = 0; j <= 127; j++)
        {
            if (code[i] == a[j])
            {
                printf("%c", j);
                break;
            }
        }
    }
    return 0;
}
```

得到 flag

hgame{noW_YOu~koNw-rea1_UPx~mAG|C_@Nd~crC16}

xdmaze
打开 ida 的字符串窗口
发现



交叉引用后找到判断语句伪代码

```
{
  for ( i = 6; i <= 33; ++i )
  {
    switch ( *((_BYTE *)v11 + i) )
    {
      case '0':
        v14 += 512;
        break;
      case '1':
        v14 += 64;
        break;
      case '2':
        v14 += 8;
        break;
      case '3':
        ++v14;
        break;
      default:
        goto LABEL_8;
    }
    if ( word_404020[v14] != 0x20 || v14 > 4095 )
    {
      v9 = std::operator<<<std::char_traits<char>>(&std::cout, "Failed");
      std::ostream::operator<<(v9, &std::endl<char,std::char_traits<char>>);
      return 1;
    }
  }
  v10 = std::operator<<<std::char_traits<char>>(&std::cout, "Win");
  std::ostream::operator<<(v10, &std::endl<char,std::char_traits<char>>);
  return 0;
```

找到判断语句

```
byte_404020     db 2 dup(20h), 3Fh dup(23h), 20h, 7 dup(23h), 20h, 1FFh dup(23h)
                                        ; DATA XREF: main+1B9↑o
                db 20h, 3Fh dup(23h), 20h, 3Fh dup(23h), 2 dup(20h), 1FFh dup(23h)
                db 2 dup(20h), 3Fh dup(23h), 20h, 7 dup(23h), 20h, 1FFh dup(23h)
                db 2 dup(20h), 7 dup(23h), 20h, 1FFh dup(23h), 2 dup(20h)
                db 7 dup(23h), 20h, 7 dup(23h), 20h, 7 dup(23h), 20h, 7 dup(23h)
                db 2 dup(20h), 3Fh dup(23h), 20h, 1FFh dup(23h), 20h, 1FFh dup(23h)
                db 2 dup(20h), 1FFh dup(23h), 20h, 3Fh dup(23h), 20h, 3Fh dup(23h)
                db 20h
```

这是 word_404020 中的值

编写 c 解密代码

```
#include<stdio.h>

int main()
{
int
array[5000]={0x20,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
```

0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x20,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,

0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x20,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,

```
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x20,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x20,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
```

0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,

0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,

```
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x20,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,
0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x23,0x
23,0x23,0x23,0x20 };
    int i=0,tk=0;//tk 记录 array 值等于 0x20 的位置
    while (array[i] != '\0')
    {
        if (array[i] == 0x20)
        {
            switch (i-tk)
            {
            case 512:printf("0"); break;
            case 64:printf("1"); break;
            case 8:printf("2"); break;
            case 1:printf("3"); break;
            }
            tk = i;
        }
        i++;
    }
    return 0;
}
```

```
31201130312032032222231003011
```

拼接后得到 flag

hgame{31201130312032032222231003011}