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

union single_data
{
  unsigned int u;
  int i;
  float f;
  unsigned char c[4]; // 4 * 8 = 32 bits
};

union double_data
{
  double d;
  unsigned char c[8]; // 8 * 8 = 64 bits
};

void initializeBinaryArray(unsigned char *binary_arr, int length)
{
  for (int i = 0; i < length; i++)
  {
    binary_arr[i] = 0;
  }
}

/**
 * 2^n_max <= x となる、最大のn_maxを求める
 */
int getMaxBinaryPOW(unsigned char max)
{
  static int n_max = 0;
  double v_binary_pow = pow(2, ++n_max);

  if (v_binary_pow > (double)max)
  {
    // n_maxであるとき(2^n_max > maxのとき)
    // 2^n_max > x かつ 2^(n_max - 1) < x を満たす
    // -> n_max - 1を返す
    int result = n_max - 1;
    n_max = 0;
    return result;
  }
  else
  {
    // n_maxではない時
    return getMaxBinaryPOW(max);
  }
}

/**
 * 与えられた数値x(unsigned char)に対する、
 * バイナリ文字列を表す配列(binary_arr)に値をセットする
 * ex) binary_arr[8] = {0,1,1,0,1,1,0,0}
 */
void setBinaryArray(unsigned char *binary_arr, unsigned char x)
{
  int n_max = getMaxBinaryPOW(x); // 2^nについて、x以下で最大の値となるnを求める

  if (x > 0)
```

```
  {
    // x = 0でなければ、tmp_binary_arrの該当の桁に1をセットする
    binary_arr[n_max] = 1;

    // xから2^n_maxを引く
    x -= (unsigned char)pow(2, n_max);

    // x = 0になるまで繰り返す
    setBinaryArray(binary_arr, x);
    return;
  }

  return;
};

void printActualBitPatterns(int data_size, unsigned char *data)
{
  int bits_length = 8 * data_size;

  // dataに対応するバイナリ文字列を表す配列
  unsigned char result_binary_arr[bits_length];

  // binary_arrをすべて0で初期化
  initializeBinaryArray(result_binary_arr, bits_length);

  // unsigned char[]1要素分 = 8bit分のbinary_arrを定義
  unsigned char tmp_binary_arr[8];
  int separation_num = data_size;

  // 8bitずつbinaryArrayを求める
  for (int i = 0; i < separation_num; i++)
  {
    // binary_arr初期化
    initializeBinaryArray(tmp_binary_arr, 8);

    // data[i]の値に対応するbinary_arrをセット
    setBinaryArray(tmp_binary_arr, data[i]);

    for (int j = 0; j < 8; j++)
    {
      //セットされたbinary_arrをresult_binary_arrに記録
      result_binary_arr[i * 8 + j] = tmp_binary_arr[j];
    }
  }

  // result_binary_arrayを逆順に表示
  // dataに対応するbinary patternが出力される
  for (int i = bits_length - 1; i >= 0; i--)
  {
    printf("%d", result_binary_arr[i]);
    i % 8 == 0 && printf(" ");
  }

  printf("\n\n");
}

int main()
{

  int n = 0;
```

```
120   union single_data sx;
121   union double_data dx;
122
123   //それぞれ初期化
124   dx.d = 0;
125   sx.i = 0;
126
127   /** unsigned int */
128   sx.u = 20;
129   printf("unsigned integer: \n");
130   printActualBitPatterns(sizeof(sx.u), &(sx.c)); //バイナリパターンを出力
131
132   /** int */
133   sx.i = 20;
134   printf("integer: \n");
135   printActualBitPatterns(sizeof(sx.i), &(sx.c)); //バイナリパターンを出力
136
137   /** float */
138   sx.f = 20;
139   printf("float: \n");
140   printActualBitPatterns(sizeof(sx.f), &(sx.c)); //バイナリパターンを出力
141
142   /** double */
143   dx.d = 20;
144   printf("double: \n");
145   printActualBitPatterns(sizeof(dx.d), &(dx.c)); //バイナリパターンを出力
146
147   return 0;
148 }
```