問題 1. コード 1 は、二次元での極座標から直交座標への変換と直交座標から極座標への変換

$$\begin{cases} x = r\cos\theta \\ y = r\sin\theta \end{cases} \qquad \begin{cases} r = \sqrt{x^2 + y^2} \\ \theta = \arctan\dfrac{y}{x} \end{cases}$$

を、それぞれ、C の関数 `ptoo()` と `otop()` として実装したものである。いずれの関数も、第一引数の指す構造体の値が変換元であり、変換結果を第二引数の指す構造体に代入する。

(1) 空欄を埋めよ。

ヒント C の標準ライブラリ関数 `hypot(`x`,` y`)` と `atan2(`y`,` x`)` は、それぞれ、$\sqrt{x^2 + y^2}$ と $\arctan(y/x)$ を計算する。`atan2()` の引数の順序に注意。

コード 1 極座標から直交座標への変換（二次元）

```
#include <math.h>

struct polar { /* 極座標 */
        double  r;      /* r 座標 */
        double  theta;  /* θ 座標 */
};
struct ortho { /* 直交座標 */
        double  x;      /* x 座標 */
        double  y;      /* y 座標 */
};

void
ptoo(const struct polar *p, struct ortho *o)
{
        (あ) = (い) * cos( (う) );
        (え) = (お) * sin( (か) );
}

void
otop(const struct ortho *o, struct polar *p)
{
        (き) = hypot( (く) , (け) );
        (こ) = atan2( (さ) , (し) );
}
```

問題 2. コード 2 は、配列の先頭を指すポインタと大きさを受け取って要素の最小の値を返す関数 `min_value` を定義する C のコードである。配列内のデータの並び順について何も仮定されない。空配列に対しては `long` で表現される最大の整数値を返す。

1. 空欄を埋めよ。
2. 関数 `min_value` の第二引数の型を `size_t` ではなく `unsigned int` にするとどのような不都合があり得るか解説せよ。
3. 配列の大きさを n として、この関数の時間計算量を評価せよ。

コード 2 ソートされていない配列の要素の最小値

```
#include <stdlib.h>
#include <limits.h>

long
min_value(const long *array, size_t size)
{
        long    m = LONG_MAX; /* 仮の最小値 */
        const long      *p;
        for (p = (す); p < (せ); p ++) {
                if ((そ) < (た)) {
                        m = (ち);
                }
        }
        return m;
}
```

問題 3. 単方向リストのセルを表す構造体 `struct cell` がファイル `slist.h` 内でコード 3 の通り定義されている。

コード 4 は、単方向線形リスト（ダミーセルなし）の先頭を指すポインタを受け取ってリストの要素中で最小の値を返す関数 `min_value` を定義する C のコードである。データの並び順について何も仮定されない。空リストに対しては `long` で表現される最大の整数値を返す。

1. 空欄を埋めよ。
2. データ構造を単方向線形リスト（先頭にダミーセル一つ）に変更する場合、コードのどこをどう修正すればよいか説明せよ。
3. リストの長さ（セルの個数）を n として、この関数の時間計算量を評価せよ。

コード 3 単方向線形リストに関する型定義

```
struct cell {
        struct cell     *next; /* 次のセルを指すポインタ */
        long            value; /* このセルに格納された値 */
};
```

コード 4 順序なし単方向リストの要素の最小値

```
#include <stdlib.h>
#include <limits.h>
#include "slist.h"

long
min_value(const struct cell *first)
{
        long    m = LONG_MAX; /* 仮の最小値 */
        const struct cell       *p;
        for (p = (つ); p != NULL; p = (て)) {
                if ((と) < (な)) {
                        m = (に);
                }
        }
        return m;
}
```

問題 4. 二分木の節点を表す構造体 `struct node` がファイル `bintree.h` 内でコード 5 の通り定義されている。

コード 6 は、二分木の根を指すポインタを受け取って節にラベルづけられた最小の値を返す関数 `min_value` を定義する C のコードである。データの並び順について何も仮定されない。空木に対しては `long` で表現される最大の整数値を返す。

1. 空欄を埋めよ。
2. 二分木の大きさ（節の個数）を n として、この関数の時間計算量を評価せよ。

コード 5 二分木に関する型定義

```
struct node {
        struct node     *left;  /* 右側の子節を指すポインタ */
        struct node     *right; /* 右側の子節を指すポインタ */
        long            value;  /* この節にラベルづけられた値 */
};
```

コード 6 順序づけられない二分木のラベルの最小値

```
#include <stdlib.h>
#include <limits.h>
#include "bintree.h"

static long
min_value_sub(const struct node *p)
{
        long    m = p->value; /* 仮の最小値 */
        if (p->left != NULL) {
                long    m_left = min_value_sub( (ぬ) );
                if ( (ね) < (の) ) {
                        m = (は) ;
                }
        }
        if (p->right != NULL) {
                long    m_right = min_value_sub( (ひ) );
                if ( (ふ) < (へ) ) {
                        m = (ほ) ;
                }
        }
        return m;
}

long
min_value(const struct node *root)
{
        return root != NULL ? min_value_sub(root) : LONG_MAX;
}
```

ヒント. C の構文　式$_1$? 式$_2$: 式$_3$　は、まず　式$_1$ を計算し、その値が真ならば 式$_2$ を計算してその結果を全体の結果とし、偽ならば 式$_3$ を計算してその結果を全体の結果とする式である。

問題 5. アルゴリズムとデータ構造について、以下の問いに答えよ。

(1) 実用的な三つのソートアルゴリズムであるクイックソート・ヒープソート・マージソートをどう使い分ければ良いか、時間計算量と空間計算量（計算時間とメモリ消費量）の観点から解説せよ。

(2) 平衡木とは何かを説明せよ。さらに、順序づけられたデータの格納に単純な順序つき二分木では不都合でなんらかの平衡木が必要になる理由を、時間計算量と空間計算量（計算時間とメモリ消費量）の観点から解説せよ。