解答例

問題1のみ （問題2～4は講義ノートを見てください）

(1) すべての要素が異なる配列（長さ n）の k 番目の要素を平均時間計算量 O(n) で計算するプログラムを作成せよ （注意：ソートするプログラムではない）。
Randomized quick sort を修正して再帰的呼び出しをするなら以下のようなプログラムが可能。 0-origin index に注意。

```
int kth_rec(int *target, int left, int right, int k) {
    int i;
    while(left < right){
        i = partition(target, left, right);
        if(k == i){
            return target[i];
        }else if(k < i){
            return kth_rec(target, left, i, k);
        }else{
            return kth_rec(target, i+1, right, k);
        }
    }
    return target[i];
}
```

講義ノートと同じ `partition` を使う。

(2) 長さ n の配列から k番目の要素を選択する時間を $C(n)$ と置けば以下のような漸化式が得られる。pivot で分けられた2つの領域の大きい方を探す可能性があるため、 max と不等式を使用している。講義ノートを参照して $\mathrm{C}(n) \in O(n)$ を解いてください。

$C(0) = 0, C(1) = 0$

$$C(n) \leq n - 1 + \frac{1}{n} \sum_{k=0,\ldots,n-1} \max\{C(k), C(n-k-1)\}$$

(3) 長さが $n = 2^{40}$ のように長い配列に対しても、stack サイズが通常の 1MB 程度で stack overflow を起こさずに動くように、プログラムを工夫せよ。

(1) のプログラムでは、target が巨大配列だと、たとえば毎回最大値が pivot に選ばれると、stack overflow する可能性がある。Stack overflow を避けるには、たとえば再帰的呼び出しを避ければよい。

```
int kth(int *target, int len, int k){
    if(k < 0 || len <= k){
       exit(1);
    }
    int left   = 0;
    int right  = len;
    int i;
    while(left < right){
        i = partition(target, left, right);
        if(k == i){
            return target[i];
        }else if(k < i){
            right = i;
        }else{
            left = i;    }
    }
    return target[i];
}
```