情報構造論　前期中間　　番号：

4EI　　(1/2)　　氏名：

５つの設問の中から、４つについて回答せよ。

1　処理速度のオーダの計算問題

N 件のデータを処理するための２つのアルゴリズムがある。

さらに、データ件数 $N = 100$ における、実際の処理時間の計測値と、それぞれのアルゴリズムの処理速度は、以下のように与えられた。

処理	$N = 100$ の処理時間	処理速度の一般式
アルゴリズム１	$T_1(100) = 100$ msec	$T_1(N) = O(N^2)$
アルゴリズム２	$T_2(100) = 200$ msec	$T_2(N) = T_a \log N + T_b N\sqrt{N} + T_c$

この時、以下の設問に答えよ。

1. アルゴリズム２の処理速度をオーダによって表せ。

2. データ件数 $N = 1000$ におけるアルゴリズム１の処理時間を大まかに予想せよ。

3. データ件数に応じて２つのアルゴリズム選択する場合、どちらのアルゴリズムが最適か、具体的な数値を交えて説明せよ。
   ただし T_a と T_b は、極端に大きさの異なる値ではない。

2　処理速度のオーダの説明問題

トランプのカードの様に扱えるデータ構造があった場合、その並べ換えに要する時間は、データ件数を n とした場合、最大でも $O(n \log n)$ により行えることを、説明せよ。

トランプのカードでは、データ列の中間へのデータ挿入が一定時間で行え、かつ並べ換えの終ったデータ列は２分探索が可能なことを意味する。

3　再帰方程式

以下の、再帰呼出しによって記述された、２分探索プログラムの処理速度のオーダを求めたい。

```
#include <stdio.h>
int bin_search( int a[] , /* 検索対象の配列 */
                int key , /* 検索するデータ */
                int s ,   /* 検索の先頭     */
                int e     /* 検索の末尾+1   */ )
{
    if ( e - s == 1 ) {  /* 対象データが１件の時だけ比較する */
        if ( a[ s ] == key )
            return s ;  /* 見つかった   */
        else
            return -1 ; /* 見つからない */
    } else {
        int  m = (s + e) / 2 ;
        if ( a[ m ] > key )
            return bin_search( a , key , s , m ) ; /* 前半から探す */
        else
            return bin_search( a , key , m , e ) ; /* 後半から探す */
    }
}

int array[] = { 1, 3, 12, 40, 100, 123, 234, 1000 } ;

void main() {
    printf( "%d\n" , bin_search( array , 123 , 0 , 8 ) ) ;
    /* 見付けた場所 5 が表示される。*/
}
```

ただし、このプログラムでは出題および簡略化の都合上、データ件数 $N = $ `e-s` には 2 のべき乗の値 $N = 2^m$ を与える。そうしないと正しく動作しない。

1. 対象となるデータ件数 N を用いて、関数`bin_search()`の処理速度 $T_{bin}(N)$ を、再帰方程式によって示せ。

2. 再帰方程式を解いて、処理速度のオーダを示せ。厳密な証明でなくても良い。

4　オブジェクト指向の真似

```
#include <stdio.h>

struct Que {           /* 手抜きの実装 */
    int  data[ 10 ] ;
    int  wp , rp ;
} ;
void QueInit( struct Que* q ) {
    q->wp = q->rp = 0 ;         /* 初期化 */
}
void QuePut( struct Que* q , int data ) {
    q->data[ q->wp++ ] = data ; /* データの追記 */
}
int  QueGet( struct Que* q ) {
    return q->data[ q->rp++ ] ; /* データの取出し */
}

void main() {
    struct Que q1 ;  /* 注意：q1,q2 を混同しないでね */
    struct Que q2 ;

    QueInit( &q1 ) ;      QueInit( &q2 ) ;

    QuePut( &q1 , 10 ) ;  QuePut( &q1 , 20 ) ;
    QuePut( &q2 , 111 ) ; QuePut( &q2 , 222 ) ;

    printf( "%d\n" , QueGet( &q1 ) ) ;
    printf( "%d\n" , QueGet( &q2 ) ) ;
    printf( "%d\n" , QueGet( &q1 ) ) ;
}
```

1. このプログラムの実行結果について述べよ。

2. このプログラムで採用されているデータ構造は、
   (a) データの書込み／取出しの順序関係から、一般的にどの様に呼ばれるか答えよ。
   (b) どの様な処理に応用されるのか、事例を交えて説明せよ。

5　再帰呼出しと処理回数

```
#include <stdio.h>
int  stack[ 100 ] ;
int  sp = 0 ;
                          /* 1行に書くのは美しくない! */
void push( int x ) { stack[ sp ] = x ; sp++ ; }
int  pop()         { sp-- ; return stack[ sp ] ; }

void foo() {
    if ( stack[ sp - 1 ] <= 1 ) {
        pop() ; push( 1 ) ;
    } else {
        int  ans ;
        push( stack[ sp - 1 ] - 1 ) ;
        foo() ;
        push( stack[ sp - 2 ] - 2 ) ;
        foo() ;
        ans = pop() + pop() ;
        pop() ;
        push( ans ) ;
    }
}
void main() {
    push( 3 ) ;
    foo() ;
    printf( "%d\n" , pop() ) ;
}
```

1. このプログラムの実行結果を示し、

2. さらに関数 `foo()` の呼出される回数を答えよ。