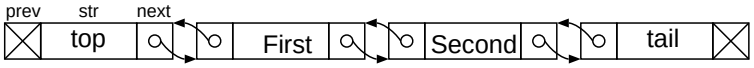
情報構造論　後期中間試験　　番号：

4EI　　(1/3)　　氏名：

5 問 回答せよ。全設問回答時は、最低点を 1/3 で加算、 100 点以上は切捨て。

1　双方向リスト

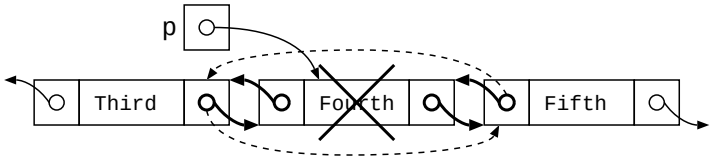
双方向リストの理解確認のために、以下のイメージ図の様なデータ構造を完成させるための命令を記述し、さらにそのデータ構造の一部を表示する命令の表示結果を述べよ。エラーの場合には『エラー』と明記すること。




```
struct BDList {
    char*          str ;
    struct BDList* prev ;
    struct BDList* next ;
} ;
struct BDList* bdlist( char* s ,
                       struct BDList* p , struct BDList* n )
{   struct BDList* nn ;
    nn = (struct BDList*)malloc( sizeof( struct BDList ) ) ;
    if ( nn != NULL ) {
        nn->str  = s ;  nn->prev = p ;  nn->next = n ;
    }
    return nn ;
}
void main() {
    struct BDList* top    = bdlist( "top" ,   NULL , NULL ) ;
    struct BDList* tail   = bdlist( "tail" ,  NULL , NULL ) ;
    struct BDList* first  = bdlist( "First" , top ,  NULL ) ;
    struct BDList* second = bdlist( "Second", first , tail ) ;
    /* 上図の様な 4 件のデータを完成させるための処理を記述 */

    ----------------------------------------

    ----------------------------------------

    ----------------------------------------
    /* 上図の様なデータで以下の処理結果は？ */
    printf( "%s\n" , tail->str ) ;                ---------------

    printf( "%s\n" , top->next->next->str ) ;     ---------------

    printf( "%s\n" , top->next[2]->str ) ;        ---------------

    printf( "%c\n" , tail->prev->str[ 2 ] ) ;     ---------------

    printf( "%d\n" , top->prev ) ;                ---------------
}
```

2　双方向リストのデータ削除

前設問と同じデータ構造で、ポインタp の指し示す 1 要素を削除する関数bddel()を作成せよ。ただしデータstr については削除する必要がない。




```
void bddel( struct BDList* p ) {
    /* この内部を作成せよ */
}
```

3　2分木構造

```
struct Tree {    /* 整数データの２分木 */
    int          data ;
    struct Tree* left ;  /* 左の枝 */
    struct Tree* right ; /* 右の枝 */
} ;
struct Tree* tree( int x ,
                   struct Tree* l , struct Tree* r )
{   /* 新しい木の節を作成する */
    struct Tree* n ;
    if ( (n = (struct Tree*)malloc( sizeof( struct Tree ) )
              != NULL ) {
        n->data = x ;  n->left = l ;  n->right = r ;
    }
    return n ;
}
void print( struct Tree* p )
{   if ( p != NULL ) {
        print( p->right ) ; /* 順番に注意 */
        printf( "%d " , p->data ) ;
        print( p->left ) ;  /* 順番に注意 */
    }
}
void main() {
    struct Tree* top
        = tree( 1 , tree( 2 , tree( 3, NULL, NULL ) , NULL ) ,
                    tree( 4 , NULL , NULL ) ) ;
    print( top ) ;
}
```

できあがったtop以下のデータ構造のイメージ図と、関数print()を実行した時の処理結果を示せ。ただし、再帰呼び出しの実行順序が判るような説明を併記すること。

4　2 分木による式の表現

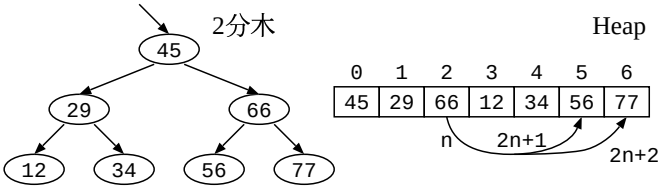
前設問のデータ構造 Tree を用いて整数と演算子だけの式の表現を行いたい。ただし整数データの場合は、 left,right 共に NULL を代入し、演算子の場合には、演算子の番号を data 部に保存する。

以下に示すプログラム中の関数 eval は、式の意味に応じた計算を実行しその答えを求める関数とする。`eval()` 内部を完成させよ。

```
/* 構造体の宣言,tree() 等の関数は、前設問の継続 */
#define PLUS  0
#define MINUS 1
#define MUL   2
#define DIV   3
int eval( struct Tree* p ) {
    if ( p == NULL ) {
        return 0 ;
    } else if ( p->left == NULL && p->right == NULL ) {
        -------------------
    } else {
        :
    }
}
void main() {
    struct Tree* exp   /* 式: 2*3 + 4/2 */
        = tree( PLUS , tree( MUL , tree( 2 , NULL , NULL ) ,
                                   tree( 3 , NULL , NULL ) ) ,
                       tree( DIV , tree( 4 , NULL , NULL ) ,
                                   tree( 2 , NULL , NULL ) ) ) ;
    printf( "%d\n" , eval( exp ) ) ; /* 8 が表示されること */
}
```

5　説明問題 1

下図に示す、２分探索木と配列によるヒープを比較して、それぞれの方式の利点・欠点を対比的に説明せよ。



6　説明問題 2

２分探索木で N 件のデータがあった場合、成長した木の形状の違いから、探索速度が最短の場合と最悪の場合について、

1. 形状の違いが発生する理由と、

2. それぞれの探索速度のオーダ記法とその算出理由

について図や式などを用いて具体的に説明せよ。