情報構造論　後期中間試験 (出題：斉藤)　番号：

4EI　　2004/12/07,(1/3)　　氏名：

以下の５つの設問の中から、４つを選んで解答せよ。全問の解答がある場合は、採点の高い4 つを合計し、最も評価の低かった設問の点数の 1/3 をさらに加点する。ただし合計が 100 を超える場合は、 100 点として扱う。

1　説明問題 (× 25)

循環型の双方向リストについて、具体的な例をあげて説明せよ。説明には、以下の物が含まれていること。

1. データ構造の宣言をプログラム例
2. データ構造の各要素の意味
3. 模式図
4. どういった情報を扱う時に便利か

2　基礎問題 (逆ポーランド記法)(× 25)

```
int  stack[ 20 ] ; /* テスト紙面の都合で1行書きが多いけど */
int  sp = 0 ;      /* 普通はやらないで！                  */
void push( int x ) { stack[ sp ] = x ; sp++ ; }
int  pop()         {  sp-- ; return stack[ sp ] ; }

int srpn( char str[] )
{  int i ;
   for( i = 0 ; str[ i ] != '\0' ; i++ ) {
       if ( str[ i ] >= '0' && str[ i ] <= '9' ) {
           push( str[ i ] - '0' ) ;  /* 1桁の数字を数値に直す */
       } else {
           int x , y ;
           switch( str[ i ] ) {
           case '+' : y = pop() ; x = pop() ; push( x + y ) ; break ;
           case '-' : y = pop() ; x = pop() ; push( x - y ) ; break ;
           case '*' : y = pop() ; x = pop() ; push( x * y ) ; break ;
           case 'N' : x = pop() ; push( -x ) ; break ;
           }
        }
    }
    return pop() ;
}
void main()
{   printf( "%d\n" , srpn( "83-" ) ) ;    /* 処理(1) */
    printf( "%d\n" , srpn( "435N+*" ) ) ; /* 処理(2) */
}
```

1. プログラム先頭の`push()`,`pop()` で扱うデータ構造を英字4文字で示せ。

2. `main()` 中の処理 (1),(2) で表示される内容を、それぞれ答えよ。

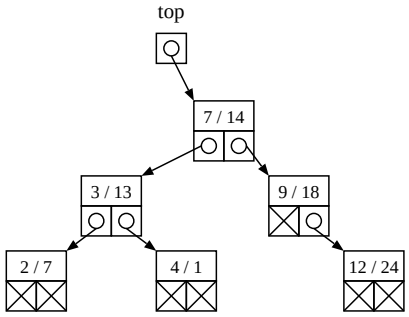
   (1) ____________

   (2) ____________

3. 一般的な計算式 `(1+2) * 3 + 4 * 5` を、このプログラムで計算するような
   逆ポーランド記法に変換せよ。

3　2 分探索木 (× 25)

月と日の誕生日のデータによる 2 分探索木のデータを作成し、その木構造から目的の誕生日が存在するかどうか検索するプログラムを作りたい。
与えられた木から、指定された月日が含まれるか探す関数 `find()` を作成せよ。



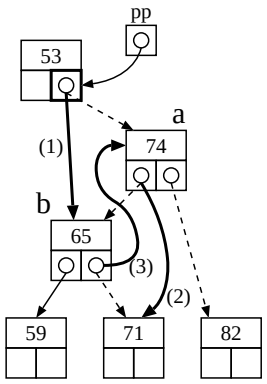

```
struct BDTree { /* 月日の 2 分木 */
    int month , day ;
    struct BDTree* left ;
    struct BDTree* right ;
} ;

int find( struct BDTree* top , int m , int d )
{   /* m,d が月日とする */
    /* 見つかったら 1, 見つからなければ 0 を返すこと */
    /* ヒント：(月 × 32 + 日) を比較せよ */
}
```

4　2 分木から AVL に (× 25)

2 分探索木で、左右の枝の大きさのバランスがとれていないと、処理速度の低下が発生する。このため、左右のバランスをとるための処理が必要となる。

右の図は、そのための処理の 1 例で 59 配下の木の深さが大きい場合に深さを 1 つ減らすための処理である。

図中の点線のポインタを太実線のように書き換える処理を記述し、`avl_SRR()` を完成させよ。



single right rotation

```
struct BTree{
    int           data ;   /* データ */
    struct BTree* left ;   /* 左枝   */
    struct BTree* right ;  /* 右枝   */
} ;
void avl_SRR( struct BTree** p )
{
    struct BTree* a = *p ;
    struct BTree* b = a->left ;
    (*p) = b ;                          /* (1) */

    ---------------------------------- /* (2) */

    ---------------------------------- /* (3) */
}
```

1. 上記プログラムの (2),(3) の部分を埋めよ。
2. 2 分探索木でバランスが取れない 2 分木ができ上がるのはどういう状況か、説明せよ。

   ヒント：2 分木にデータを追加する処理などの概要も説明すること。

5　プログラム作成 (× 25)

2 分木のデータ構造を、意志決定の質問や結論の情報を表すために応用した。いくつかの yes/no の質問の最後に、その質問から結論を表示するプログラムを以下のように作成した。

```
struct DTree {
    char*  message ;     /* 質問か結論のメッセージ */
    struct DTree* yes ;  /* yes の時の決定木 */
    struct DTree* no ;   /* no  の時の決定木 */
} ;
struct DTree* dtree( char* mes , struct DTree* y , struct DTree* n )
{   struct DTree* nn = (struct DTree*)malloc( sizeof( struct DTree ) ) ;
    if ( nn != NULL ) {
        nn->message = mes ;
        nn->yes = y ;
        nn->no  = n ;
    }
    return nn ;
}
void main()
{   struct DTree* top =
        dtree( "授業面白い？" ,
            dtree( "数式大好き？" ,
                dtree( "大学進学しよう" , NULL , NULL ) ,
                dtree( "技術職になろう" , NULL , NULL ) ) ,
            dtree( "よーく考えよう" , NULL , NULL ) ) ;
            ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~(A)
    struct DTree* p = top ;

    while( p != NULL ) {
        if ( ______________________________ ) {
                                          (B)
            printf( "結論：%s\n" , p->message ) ;
            break ;                   ~~~~~~~~~~(C)
        } else {
            char buff[ 10 ] ;
            printf( "質問:%s" , p->message ) ;
            scanf( "%s" , buff ) ;
            if ( strcmp( buff , "yes" ) == 0 ) {
                 ~~~~~~~~~~~~~~~~~~~~~~(D)
                p = p->yes ;
            } else {
                p = p->no ;
            }       ~~~~~(E)
        }
    }
}
```

1. プログラムで、`main()` 中の`top` に代入されているデータ構造の模式図を描け。

2. 下線部 (B) には、どの様な条件式を記入するのが良いか答えよ。

3. 下線部 (A),(C),(D),(E) のデータ構造の型を答えよ。