情報構造論 (斉藤)　後期中間試験　　番号：

4EI　2010/11/29　(1/3)　　氏名：

5 つの設問の中から、4 つを選んで回答せよ。各設問 25 点

1　動作トレース (25)

```
int stack[ 10 ] , sp = 0 ; // スタック
int var[ 26 ] ;            // 変数 A-Z の値保存
int rpn( char s[] ) {  int i ;
   for( i = 0 ; s[i] != '\0' ; i++ ) {
      if ( s[i] == '=' && 'A' <= s[i+1] && s[i+1] <= 'Z' ) {
         int vid = s[++i] - 'A' ;
         var[ vid ] = stack[ --sp ] ;
      } else if ( 'A' <= s[i] && s[i] <= 'Z' ) {
         int vid = s[i] - 'A' ;
         stack[ sp++ ] = var[ vid ] ;
      } else if ( '0' <= s[i] && s[i] <= '9' ) {
         int val = s[i] - '0' ;
         stack[ sp++ ] = val ;
      } else if ( s[i] == '+' ) {
         stack[ sp-2 ] = stack[ sp-1 ] + stack[ sp-2 ] ;
         sp-- ;
      } else if ( s[i] == '*' ) {
         stack[ sp-2 ] = stack[ sp-1 ] * stack[ sp-2 ] ;
         sp-- ;
      }
      if ( sp > 0 ) // スタック最上部を表示
         printf( ">%d\n" , stack[ sp-1 ] ) ;
   }
   return stack[ --sp ] ;
}
void main() {
   printf( "%d\n" , rpn( "1=A,2=B,3A+B*" ) ) ;
}
```

[設問] このプログラムの実行により出力される内容を答えよ。

解答欄

2　説明問題

1. 2 分探索木を用いたデータベースで、データ件数が 1000 件で平均検索時間が 10[msec] であった場合、100000 件では、平均検索時間はどの程度となるか答えよ。

2. ハッシュ法を用いたデータベースで、文字列をキーとして検索する場合、ハッシュ関数にはどの様な計算をするのが望ましいのか答えよ。また、その場合のデータ件数 N における処理時間のオーダを示せ。

3　2 分木の基本 (25)

```
struct Tree {                              | ★以下のプログラムの (A),(B) の行が終わった
   int          data ;                     |  時点で、top の先に生成されているデータ構造
   struct Tree* left ;                     |  のイメージ図を、それぞれ記載せよ。
   struct Tree* right ;                    |
} ;                                        | void main() {
struct Tree* top = NULL ;                  |    struct Tree* L ;
                                           |    struct Tree* M ;
void entry( int x ) {                      |    entry( 10 ) ;
   struct Tree** tail = &top ;             |
                                           |    entry( 20 ) ;
   while( *tail != NULL ) {                |    entry( 30 ) ; /* (A) */  (13)
      if ( (* tail )->data == x )          |
             ~~~~(A)                       |    L = top ;
         break ;                           |    M = top->right ;
      else if ( (*tail)->data > x )        |    L->right = M->left ;
         tail = &( (*tail)->left ) ;       |    M->left = L ;
      else          ~~~~~~~~~~~~~(B)       |    top = M ;     /* (B) */  (12)
         tail = &( (*tail)->right ) ;      | }
   }             ~~~~~~~~~~~~~~~~~~~(C)
   if ( *tail == NULL ) {
      *tail = (struct Tree*)malloc( sizeof( struct Tree ) ) ;
      if ( *tail != NULL ) {          ~~~~~~~~~~~~~~~~~~~~~~(D)
         (*tail)->data = x ;
         (*tail)->left = (*tail)->right = NULL ;
      }                    ~~~~~~~~~~~~~~~~~~~~~(E)
   }
}
```

4　データ構造の方と説明 (× 25)

1. 前設問で、下線部 (A)〜(E) の型を答えよ。(10)

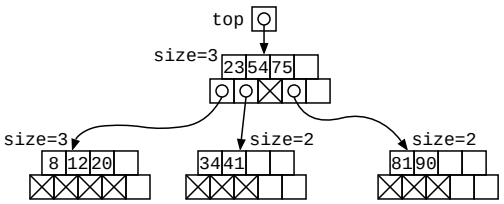
   (A)_____　(B)_____　(C)_____

   (D)_____　(E)_____

2. 前設問で、(A) ⇒ (B) のようなポインタのつなぎ替えに関係するデータ構造の名前を答え、そのつなぎ替えによって どの様な利点が得られるようにするのか、答えよ。(5+10)

5　B木

データベースなどで採用されている B 木の中からデータの有無を探すプログラム`find(ポインタ, 値)` を作成せよ。実際のデータイメージは、右図に示す。下線部 (A)〜(E) に相応しい処理を記入せよ。




```
#define SIZE 4 // ノード内の最大データ数

struct BTree {
    int           size ; // 実データ数
    int           data[ SIZE ] ;

                                    // ptr
} ; ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~(A)
// BTree 内に key が見つかれば 1 , 無ければ 0 を返す関数
int find( struct BTree* p , int key ) {
   struct BTree* n ;

   for( ;                ; p = n ) {
          ~~~~~~~~~~~~~~(B)
      int i ;
      for( i = 0 ; ~~~~~~~~~~~~~~~~~~; i++ ) {
                                    (C)
         if ( p->data[i] == key ) {

            ~~~~~~~~~~~~~~~~~~~~~~~(D)
         } else if ( p->data[i] > key ) {
            n = p->ptr[i] ;
            break ;
         }
      }
      if ( i == p->size ) {

         n =
      }      ~~~~~~~~~~~~~~~~~~~~~~~(E)
   }
   return 0 ;
}
```