情報構造論 (斉藤)　前期期末試験　　番号：

4EI　2010/07/28　(1/3)　　氏名：

5 つの選択問題より、4 個を選んで回答せよ。

1　処理トレース

```
struct List {
   int          data ;
   struct List* next ;
} ;

void print( struct List* p ) {
  for( ; p != NULL ; p = p->next )
    printf( "%d " , p->data ) ;
  printf( "\n" ) ;
}

void remove( struct List**pp , int key )
{  for( ; *pp != NULL ;  pp = &((*pp)->next) )
          ~~~~~~~~~~(A)      ~~~~~~~~~~~~~~(B)
      if ( (*pp)->data == key )
           ~~~~~~~~~~(C)
         break ;
   if ( *pp != NULL ) {
      struct List* del = *pp ;
      * pp = (*pp)->next ;
        ~~(D)
      free( del ) ;
   }
}
void main() {
  struct List* top =
     cons( 1, cons( 2, cons( 3, NULL ) ) ) ;
              ~~~~~~~~~~~~~~~~~~~~~~~~~~(E)
  print( top ) ;
  remove( &top , 2 ) ;
}
```

```
struct List* cons( int x , struct List* n ) {
   struct List* ans ;
   ans = (struct List*)malloc(sizeof(struct List)) ;
   if ( ans != NULL ) {
      ans->data = x ;
      ans->next = n ;
   }
   return ans ;
}
```

設問 1
プログラムの実行結果を答えよ (5)

設問 2
remove() 実行後の top の
イメージ図を答えよ (5)

設問 3 下線部の型を答えよ。(3x5)

(A)_____ (B)_____

(C)_____ (D)_____

(E)_____

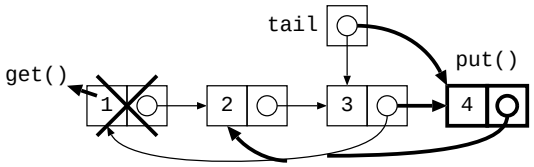
2　説明問題

以下の 2 つのデータ構造について、(a) 管理するための変数宣言と,(b) データの追加処理か取出し処理を示し、簡単にその動作を説明せよ。(12+13)

1. リストを用いた Stack(スタック)
2. 配列を用いた Queue(待ち行列)

3　プログラム穴埋め

循環リストを用いた待ち行列のプログラムを作る。
ただし、出題の簡略化のために、行列には必ず 1 件のデータが入った状態で使うものとする。
下線部に相応しい処理を埋めよ。(5x5)



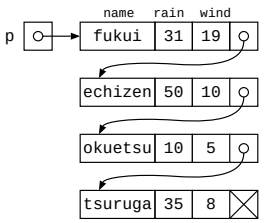

```
struct List* top ;
struct List* tail ;

void put( int x )
{
   tail->next =
                ~~~~~~~~~~~~~~~~~~~~~~~~~(A)

   tail =
          ~~~~~~~~~~~~~~~~~~~~~~~~(B)
}
```

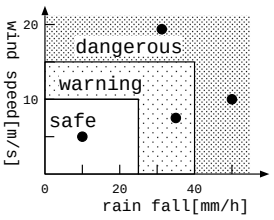
```
int get() {
   struct List* del = tail->next ;
   int ans = del->data ;

   tail->next =
                    ~~~~~~~~~~~~~~~~~~(C)

   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~(D)
   return ans ;
}
```

```
void main() {
  top = cons( 1 , cons( 2 , cons( 3 , NULL ) ) ) ;
  tail = top->next->next ;

  // 循環リストにする

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~(E)
  put( 4 ) ; // 末尾に追加
  printf( "%d" , get() ) ; // 1 を表示
}
```

4　応用問題

地域名, 雨量 [mm/h], 風速 [m/s] のデータをリスト構造で保存してある。関数の引数にリスト構造の先頭ポインタを渡すと、その雨量, 風速に応じて警告メッセージ (dangerous,warning,safe) を出力したい。
このデータを扱うために相応しい宣言と、各地域名に応じた警告メッセージを出力する関数を作成せよ。





5　リストによる多倍長計算

大量の桁数の計算ができるように、リスト構造１要素に最大 1000 までの値を保存する。リストには、計算がしやすいように、下の桁から 3 桁分づつデータが並んでいる。下の`main()` では、999999 + 000001 を計算する。

また回答が簡単になるように、リストの`p1,p2` の要素数は必ず同じ件数とする。

この大量桁の加算を行う`bignum_add()` と、リストの数値を 3 桁づつ逆順に表示する`print_rev()` を完成させよ。

```
#define MAX 1000
struct List* bignum_add( struct List* p1 , struct List* p2 )
{  struct List*  ans  = NULL ;

   struct List** tail = _____________________;
                        ~~~~~~~~~~~~~~~~~~~~(A)3
   int carry ; // 桁上がり
   for( carry = 0 ;
        p1 != NULL   && p2 != NULL ;
        p1 = p1->next , p2 = p2->next ) {

      int sum = ____________________________________________________________(B)3
                ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

      carry =
              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~(C)3
      sum   = sum % MAX ;

      *tail =
              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~(D)3

      tail  =
              ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~(E)3
   }

   if ( _________________ )
        ~~~~~~~~~~~~~~~~~(F)3
      *tail = cons( carry , NULL ) ;
   return ans ;
}

void print_rev( struct List* p ) {

   if ( _________________ ) {
        ~~~~~~~~~~~~~~~~~(G)3


      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~(H)4
      printf( "%03d " , p->data ) ;
   }
}

void main() {
   struct List* a = cons( 999 , cons( 999 , NULL ) ) ;
   struct List* b = cons( 1 ,   cons( 000 , NULL ) ) ;
   print_rev( bignum_add( a , b ) ) ;
}
```