情報構造論　前期期末　　番号：

4EI　　(1/3)　　氏名：

★の付いた７つの設問から、５つを選択し回答せよ。

それ以上の回答がされている場合は、1/3 にて加算する。設問中 `#include<...>` は、省略する。

1　ヒープメモリを使った穴埋め★

複素数の加算乗算を行う関数を、malloc を用いて作成した。

1. プログラム中の下線部 [A],[D],[E],[F],[H] を埋めよ。
2. プログラム中の下線部 [B] の判別を行う理由を述べよ。

```
struct Complex {
    double  re ;
    double  im ;
} ;
struct Complex* add( struct Complex* p1 ,
                     struct Complex* p2 )
{   struct Complex* ret =
        (struct Complex*)malloc( sizeof( _________________ ) ) ;
    if ( ret != NULL ) {                                [A]
         ~~~~~~~~~~[B]
        ret->re = p1->re + p2->re ;
        ret->im = p1->im + p2->im ;
    }             ~~~~~~~~~~~~~~~[C]
    return ret ;
}
struct Complex* mul( struct Complex* p1 ,
                     struct Complex* p2 )
{   struct Complex* ret =
        ____________________________________________________ ;
    if ( ret != NULL ) {                                  [D]
        ret->re = ___________________________________ ;
                                                    [E]
        ret->im = ___________________________________ ;
    }                                               [F]
    return ret ;
}
struct Complex c_1 = { 1 , 0 } ;
struct Complex c_j = { 0 , 1 } ;
void main() {
    struct Complex* c_1_j = add( &c_1 , &c_j ) ;
                                  ~~~[G]
    _______________ c_ans = mul( c_1_j , &c_j ) ;
               [H]
    printf( "%f + j*%f\n" , c_ans->re , c_ans->im ) ;
}
```

2　集合としてのリスト演算★

数値データのリスト構造を用いて、1 ～ 100 までの素数を求めたい。

指定された数 x が、素数列 p のデータで割り切れるものがあったら 0 、割り切れなかったら 1 を返す関数 is_prime() を作成せよ。

```
struct List {
    int          data ;
    struct List* next ;
} ;
struct List* cons( int d , struct List* n )
{   struct List* ret = (struct List*)malloc( sizeof( struct List ) ) ;
    if ( ret != NULL ) {
        ret->data = d ;
        ret->next = n ;
    }
    return  ret ;
}
int is_prime( int x , struct List* p )
{
    /* この部分を作成せよ */









}
void main()
{   int  i ;
    struct List* prime = NULL ;
    for( i = 2 ; i < 100 ; i++ ) {
        if ( is_prime( i , prime ) )  /* 素数ならリストの先端に加える */
            prime = cons( i , prime ) ;
    }
    for( ; prime != NULL ; prime = prime->next )
        printf( "%d\n" , prime->data ) ;
}
```

3　リスト基本

複数の数値データを**配列**と**リスト**によって記憶する。データは小さいデータから大きなデータへと昇順で記憶するために、既に記憶しているデータ列に新たなデータを追加する関数array_insert(),list_insert() をそれぞれ作る。

3.1　リストの場合★

リスト処理プログラムに対して、データ列を 10 2 7 の順序で与えるものとする。この時 list_insert() が呼び出される度にどの様にリストが延びていくかを、移り変わる様子が分かるように図によって示せ。

```
struct List {
    int          data ;
    struct List* next ;
} ;
struct List* cons( int d , struct List* n )
{   struct List* ret = (struct List*)malloc( sizeof( struct List ) ) ;
    if ( ret != NULL ) {                          ~~~~~~~~~~~~~~~~~~~~~~[A]
        ret->data = d ;
        ret->next = n ;
    }   ~~~~~~~~~[B]
    return  ret ;
}
struct List* list_insert( struct List* p , int x )
{   struct List** pp ;
    for( pp = &p ; *pp != NULL ; pp = &( (*pp)->next ) ) {
        if ( (*pp)->data > x )              ~~~~~[C]
            break ;
    }
    *pp = cons( x , *pp ) ;
    return p ;
}
void main()
{   int          d ;
    struct List* top = NULL ;
    while( scanf( "%d" , &d ) == 1 ) {
        top = list_insert( top , d ) ;
    }         ~~~~~~~~~~~~~~~~~~~~~~[D]
    :
}
```

3.2 配列の場合★

同じ処理を配列によって記述した array_insert() の内部を完成させよ。

```
void array_insert( int p[] , int* psize , int x )
{
    /* x より大きいデータをみつけたら、 */
    /* その場所以降を 1 つ後ろにコピー  */
    /* 空けた場所に x を入れる          */
    /* データ件数 *psize を 1 つ増加    */









}
void main()
{
    int  array[ 1000 ] , size = 0 ;
    int  d ;
    while( scanf( "%d" , &d ) == 1 ) {
        array_insert( array , &size , d ) ;
    }
     :
}
```

3.3 処理速度の理解★

データが N 件入力させた時点で、それぞれ list_insert(),array_insert() の処理の速度のオーダを述べ、簡単にその理由を述べよ。

4　プログラム中のデータの型★

これ以前の設問の中の下線で示される式の型を述べよ。

- 設問 1 [C],[G]

- 設問 3.1 [A],[B],[C],[D]

5　応用問題★

トランプのポーカーのプログラムを作りたい。

カードデータは下に示す様なリスト構造で扱う。要素 suit は、'S': スペード,'H': ハート,'D': ダイヤ,'C': クラブに対応づける。要素 rank は、カードの数字とし、エースはカードの優劣判定のために、 14 として扱う。

役が成立しているか判定する関数を、何か 1 つについて作成せよ。下の例では、`is_flush()` という関数名で記入した。

ただし問題を簡単にするため、カードは昇順に並べられているものとする。

```
#define JAC 11
#define QEN 12
#define KNG 13
#define ACE 14

struct CardList {
    char             suit ;
    int              rank ;
    struct CardList* next ;
} ;
```

```
struct CardList* card_cons( char s , int r , struct CardList* n )
{   struct CardList* ret =
        (struct CardList*)malloc( sizeof( struct CardList* ) ) ;
    if ( ret != NULL ) {
        ret->suit = s ;
        ret->rank = r ;
        ret->next = n ;
    }
    return  ret ;
}
int is_flush( struct CardList* p )
{
    /* こういった関数を作る */

}
void main()
{
    struct CardList* card =
        card_cons( 'H' , 2 ,
          card_cons( 'H' , 5 ,
            card_cons( 'H' , 8 ,
              card_cons( 'H' , JAC ,
                card_cons( 'H' , KNG , NULL ) ) ) ) ) ;
    if ( is_flush( card ) )
        printf( "フラッシュ成立\n" ) ;
}
```