以下の 5 つの設問の中から、4 つを選んで回答せよ。

1　基本動作トレース (25)

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

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 ;
}

struct List* prime_factorize( int x ) {
   struct List* ans = NULL ;
   struct List** tail = &ans ;
   for( int i = 2 ; i <= x ; i++ ) {
                  ~~~~~~(D)
      for( ; x % i == 0 ; x /= i ) {
         *tail = cons( i , NULL ) ;
         tail = &( (*tail)->next ) ;
                 ~~~~~~~~~~~~~~~~(E)
      }
   }
   return ans ;
}
```

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

int main() {
   struct List* a = prime_factorize( 2 ) ;
                    ~~~~~~~~~~~~~~~~~~~~(C)
   struct List* b = prime_factorize( 12 ) ;
   print_list( a ) ; /* (a) */
   print_list( b ) ; /* (b) */
}
```

(設問 1)
main 関数の中の 2 つの print_list() で表示される内容を答えよ。

(a) _____(7)

(b) _____(8)

(設問 2) 下線 (A)〜(E) の型を答えよ

(A)_____ (B)_____

(C)_____ (D)_____

(E)_____ (2x5)

2　説明問題 (25)

以下の 3 つの説明問題から 2 つを選び答えよ。

1. C 言語の typedef 命令について例を交えて説明せよ。
2. C 言語のビットフィールドについて例を交えて説明せよ。
3. リストを使った FIFO 型のデータ構造について、イメージ図もしくはプログラムコードを交えて説明せよ。

3　説明問題 (25)

下線部 (A)〜(H) を穴埋めせよ。回答欄 (B),(C),(D),(E) はオーダー記法で回答すること。

1. リスト構造と配列を比べた場合、配列の中から目的のデータを探すのにかかる時間は、あらかじめデータを昇順に並べておく (A)_____法を用いれば、検索に要する時間は、データ件数を N としたとき (B)_____となる。しかしリスト構造であれば、先頭から 1 づつ探すことから処理時間は、(C)_____となる。(4+3+3)

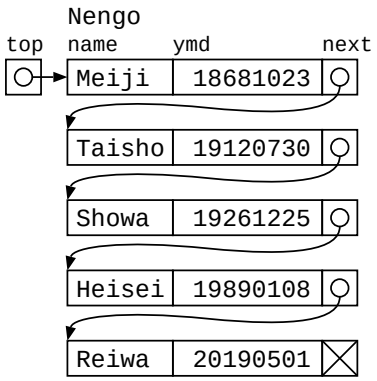
2. 一方で、配列では昇順のデータ列の途中にデータを挿入する場合、データを挿入すべき場所を見つけた後に、データを挿入する場所を確保するためにデータを 1 つずらす処理が必要となる。この処理には (D)_____の時間がかかる。これに対して、リスト構造であれば、データの挿入場所を見つけた後の挿入には (E)_____の時間がかかる。(3+3)

3. 最大でもデータ件数が 300 件、平均約 200 件の整数型のデータを保存する場合、配列であれば、このデータを覚えるのにメモリは (F)_____byte 必要となる。リスト構造の場合は `sizeof( struct List )` は、1 件あたり (G)_____byte なので、このデータ全体を覚えるには (H)_____byte を必要とする。(3+3+3)

   ただし、`int` 型は 32bit とし、`sizeof(struct List*)` は 8 とする。ヒープ領域の管理に必要なメモリについては考慮しなくていい。

4　プログラム作成 (25)

西暦の年月日のデータを和暦に変換して出力するプログラムを作りたい。西暦年月日は、2023/8/3 は 20230803 の様に 8 桁の 10 進数で扱うものとする。
右図に示すように明治以降の和暦の始まった西暦, 年, 月, 日をリスト構造で保存し、このリストを使って入力された西暦を和暦に表示して表示したい。(参考: 入力と出力の例)



1. 右図にしめす構造体 `NengoList` を定義し、
2. 西暦を和暦に変換し表示する関数 `print_wareki()` を作成せよ。

```
struct NengoList {
```

```
| int main() {
|    struct NengoList* top = リスト生成処理 ;
|    int ymd ;
|    while( scanf( "%d", &ymd ) == 1 )
|       print_wareki( ymd , top ) ;
|    return 0 ;
| }
|   (注意) リスト生成処理は作らなくていい。(10+15)
| 入力と出力の例
| 19261224 => Taisho 15,12,24
| 19261225 => Showa 1,12,25
| 20230805 => Reiwa 5,8,5
```

5　プログラム穴埋め問題 (25)

30 以下の自然数の集合を (a) 集合の要素をリストとしたもの、(b) 集合の要素を配列に格納し配列の最後には-1 を保存したもの、(c) 集合の要素に対応した 2 進数 (最下位 bit を 0 とする) で扱う場合、(a) 方式と (b) 方式の集合積を (a) 方式で求める関数 and_list_array()、(c) 方式と (a) 方式の集合和を (c) 方式で求める関数 or_bits_list() を作成する。

以下のプログラムの下線部にふさわしい処理を記載せよ。

```
struct List* and_list_array( struct List* p ,        array       ) {
                                                  ~~~~~~~~~~~~~~~~~~(A)3
   struct List* ans = NULL ;
   for( ; p != NULL ;                    ) {
      int i ;          ~~~~~~~~~~~~~~~~(B)3
      for( i = 0 ; array[i] >= 0 ; i++ )
         if ( array[i] ==              )
            break ;       ~~~~~~~~~~~(C)3
      if ( array[i] >= 0 )
         ans =                                ;
   }           ~~~~~~~~~~~~~~~~~~~~~~~~~~~~(D)3
   return ans ;
}
int or_bits_list(          bits ,                p ) {
                  ~~~~~~~~~(E)3   ~~~~~~~~~~~~~~~(F)3
   int ans = bits ;
   for( ;             ; p = p->next )
         ~~~~~~~~~~~~(G)3
      ans
         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~(H)4
   return ans ;
}
```

```
void main() {
   struct List* a =
       cons( 1, cons( 3, cons( 5, NULL ) ) ) ;
   int b[] = { 3 , 6 , 9 , -1 } ;
   int c = (1<<1) | (1<<2) | (1<<3) ;
   print_list( and_list_array( a , b ) ;
   print_bits( or_bits_list( c , a ) ;
}
```

(注意)
List や print_list() は
設問 1 の内容を引き継ぐ
(上記例の結果)
print_list() - 3
print_bits() - 1,2,3,5

```
void print_bits( int bits ) {
   for( int i = 0 ; i <= 30 ; i++ )
      if ( (1 << i) & bits )
         printf( "%d " , i ) ;
   printf( "\n" ) ;
}
```