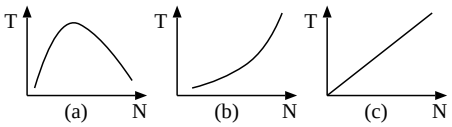
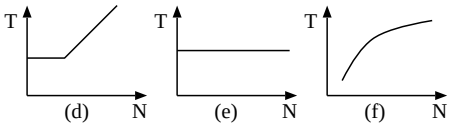
6 つの設問の中から 5 つを選んで回答せよ。

1　基本問題 (× 20)

電話番号のデータを保存し、登録されているか検索するプログラムを作りたい。

電話番号は、6桁の数値で0は登録させない。
このプログラムを右に示す3つの方式で作るとして、以下の設問に答えよ。





グラフ概形(縮尺は一定でない)

(方式1)
```
int table[ MAXSIZE ] ;
int size ; /* データ件数 */
```
電話番号は配列に昇順に保存。
検索は2分探索を行う。

(方式2)
```
struct PhoneList {
    int data ;
    struct PhoneList* next ;
} ;
struct PhoneList* table[ 100 ] ;
```
電話番号をチェイン法で保存。
ハッシュ関数は電話番号末尾2桁。

(方式3)
```
int table[ 1000000 ] ;
```
配列は0で初期化
```
void entry( int phone ) {
    table[ phone ] = phone ;
} ;
```
電話番号を添字とする場所に保存。

1. 3 つの方式それぞれで、データ件数 N と検索時間 T の関係の表すグラフとしてふさわしいものを (a) 〜 (f) の中から選べ。

   (方式 1)　　(方式 2)　　(方式 3)

2. データ件数が 500 件の時の平均検索時間は、それぞれ 2 分検索 (方式 1) で 10[msec] と、チェイン法 (方式 2) で 5[msec] であったとする。

   データ件数が 1000 件の場合の平均検索時間を、それぞれ大まかに予測せよ。

(方式 1)

(方式 2)

2　処理の選択 (× 20)

設問 1 において、検索時の処理速度を優先してプログラムを開発する場合、予測されるデータ件数に応じて (方式 1),(方式 2) をどう使い分けるべきか答えよ。具体的に『何件以上なら...』という件数を具体的に示すこと。件数などの数値は式を交えたものでも良い。

3　プログラム記述問題 (× 20)

設問 1 で、(方式 1),(方式 2) のどちらか一方の『データを探す処理』を具体的に記述せよ。関数名や引数は、任意に決めれば良い。

4　説明問題 (× 20)

いくつかある動的なメモリの管理方法について、1 つを選び、応用事例などを具体的に交えながら説明せよ。

5　データ構造の設計 (× 20)

音楽の歌手・曲名・演奏時間を保存し、歌手名で検索するデータベースを構築したい。

このデータを検索速度優先で扱うためのデータ構造について、データ構造宣言・変数宣言・そのデータの扱うための方法を説明せよ。

ただし、$O(N)$ のアルゴリズムは使わない事。データ構造は、プログラム言語での宣言文を交え、検索方法は、アルゴリズムの説明で良い。

6　アルゴリズムと処理速度 (× 20)

誕生日と名前のデータを管理するプログラムを以下のように作った。
kyaron の行の終了時点でのメモリの内容を図中に示せ。

```
#define TABLESIZE 5
struct NameBirthday {
    int  use ;
    int  month , day ;
    char name[ 20 ] ;
    struct NameBirthday* next ;
} ;
struct NameBirthday  table[ TABLESIZE ] ; /* データの実体   */
struct NameBirthday* freelist ;           /* フリーリスト   */
struct NameBirthday* ptr[ TABLESIZE ] ;   /* 使用中のデータ */
int    size = 0 ;

void init() { int i ;
    freelist = &table[ 0 ] ;
    for( i = 0 ; i < TABLESIZE - 1 ; i++ )
        table[ i ].next = &table[ i+1 ] ;
    table[ TABLESIZE-1 ].next = NULL ;
}
struct NameBirthday* nameBirthday( int m , int d , char s[] )
{   struct NameBirthday* p = freelist ;
    freelist = freelist->next ;
    p->month = m ; p->day = d ; strcpy( p->name , s ) ;
    ptr[ size++ ] = p ;
    return p ;
}
void gc() {  int i ;
    for( i = 0 ; i < TABLESIZE ; i++ )   /* マーク初期化 */
        table[ i ].use = 0 ;
    for( i = 0 ; i < size ; i++ )        /* 使用マーク付け */
        if ( ptr[ i ] != NULL )
            ptr[ i ]->use = 1 ;
    for( i = 0 ; i < TABLESIZE ; i++ ) { /* スイープ */
        if ( table[ i ].use == 0 ) {
            table[ i ].next = freelist ;
            freelist = &table[ i ] ;
        }
    }
}
void main() {
    struct NameBirthday* tohru   = nameBirthday( 2, 7,"t-saitoh" ) ;
    struct NameBirthday* tomoko  = nameBirthday( 3,13,"tomoko"   ) ;
    struct NameBirthday* mitsuki = nameBirthday( 7,14,"mitsuki"  ) ;
    struct NameBirthday* ayuka   = nameBirthday( 9,18,"ayuka"    ) ;
    struct NameBirthday* kyaron ;
    ptr[0] = NULL ;
    ptr[1] = NULL ;
    gc() ;
    kyaron = nameBirthday( 9,9,"kyaron" ) ;
}
```