目次

※[C/J]　… C 言語・Java を対象とした演習
　[C]　　… C 言語のみを対象とした演習

プログラム追加課題を始める前に…

各演習の指示の通りにプログラムを実装しましょう。
ただし、入力値の範囲指定があるものについては、指示通りの値で入力されているかのチェックは不要とします。

問題は簡単なものから、難しいものまであります。自分のスキルと相談しながらどの問題に着手するか選んでください。
各問題にそれぞれ解答目安時間が設定されています。できるだけ目安時間内に完成できるよう、心がけましょう。
どうしても解けない場合は解答例がありますので、講師に聞いてください。
解答例を見ながら、次は自力で完成できるよう、各行で何をしているか、どんな意図があるかを理解しながらプログラムを入力しましょう。

Java については、各クラスファイルは `jp.winschool.java.enshu` パッケージ内に作成しましょう。

演習 12 から、ソート（並べ替え）の問題が続きます。それぞれ手順は異なりますが、どれもやっていることは「昇順に並べ替える」です。なので、どのプログラムでも入力が同じであれば下記例 1 のように出力結果は同じになります。
例 1　　入力：`5 8 2 3 1 7 1 9 4 6` → 出力：`1 1 2 3 4 5 6 7 8 9`

それぞれのアルゴリズムで人間が理解しやすかったり、プログラム的に処理時間が短かったりと、特徴があります。それぞれの特性を考え、理解しながら実装してください。

企業での作業時にはソートプログラムが準備されているか、Java で準備されている `List.sort()`を使うなど、自分でソートアルゴリズムを考えて作成することはほぼありません。
しかし、物事を順序立てて考える練習としては重要なので出題しています。

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習1　年齢に応じたメッセージの出力　（15 分）

ファイル名：C 言語　…　enshu01.c　　Java　…　Enshu01.java

整数型の変数 nenrei に、初期化のタイミングで 0～100 の任意の年齢を設定し、その値に応じて下記のメッセージを出力させましょう。

ただし、「¥」および「%」は半角文字で出力させてください。

nenrei の値	出力メッセージ
0 ～ 19	未成年です。 地方自治体によっては医療費が¥200 のところも。
20 ～ 59	成年です。 飲酒・喫煙は控えめに。
60 ～ 100	定年後も元気 100%で過ごしてください。

ヒント 1

nenrei の範囲は各ケースで「0 以上で、19 以下だろうか」「20 以上で 59 以下だろうか」「60 以上で 100 以下だろうか」と全て書かなくても `else if` や `else` を使うことでもっと記述を省略できます。

ヒント 2

「¥」や「%」はプログラミング言語にもよりますが、エスケープ文字など特殊な意味を持つ記号として使われることのある文字です。

出力の際には"¥"とだけソース上で記述しても「¥」は出力されないので注意してください。

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習2　九九の表の出力　（20 分）

ファイル名：C 言語　…　enshu02.c　　Java　…　Enshu02.java

2 重ループを使い、下記の九九の表を出力させましょう。

ただし、Java の場合は数値が右揃えになっていなくても良いものとします。

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

ヒント 1

各行を出力するループと、各列を出力するループの 2 重ループが必要です。

行ごとに、全列の分を出力できたら改行させてください。

ヒント 2

C 言語の場合、%5d のように%と d の間に桁数を指定することで、その文字数分のスペースを確保した出力ができます。

例：`printf("ABC%6dXYZ", 1234);` →

A	B	C			1	2	3	4	X	Y	Z

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習3　九九の表の出力（応用）　（20 分）

ファイル名：C 言語　…　enshu03.c　　Java　…　Enshu03.java

2 重ループを使い、下記の九九の表を出力させましょう。
掛け算の結果が 2 の倍数の時は「**」、3 の倍数の時は「@@」、6 の倍数の時は「##」を掛け算の結果の代わりに表示させましょう。
ただし、Java の場合は数値が右揃えになっていなくても良いものとします。

```
      1    2    3    4    5    6    7    8    9
 1    1   **   @@   **    5   ##    7   **   @@
 2   **   **   ##   **   **   ##   **   **   ##
 3   @@   ##   @@   ##   @@   ##   @@   ##   @@
 4   **   **   ##   **   **   ##   **   **   ##
 5    5   **   @@   **   25   ##   35   **   @@
 6   ##   ##   ##   ##   ##   ##   ##   ##   ##
 7    7   **   @@   **   35   ##   49   **   @@
 8   **   **   ##   **   **   ##   **   **   ##
 9   @@   ##   @@   ##   @@   ##   @@   ##   @@
```

ヒント 1
6 の倍数は 2 の倍数でも 3 の倍数でもあるので、`else if` でチェックする順番に注意しましょう。

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習4　アルファベットの大文字・小文字変換　（20 分）

ファイル名：C 言語　…　enshu04.c　　Java　…　Enshu04.java

文字型の変数 beforeChar に初期化のタイミングで任意のアルファベット 1 文字を設定し、大文字が設定されている時は小文字に、小文字が設定されている時は大文字に変換して出力しましょう。
アルファベット以外の入力は無いものとします。
ただし、C 言語の場合は任意の 1 文字を入力させるよう実装しても良いものとします。

beforeChar の値	出力値
a	A
b	B
:	:
z	Z
A	a
B	b
:	:
Z	z

ヒント 1
「beforeChar の値が'a'ならば'A'、'b'ならば'B'、'c'ならば…」のように全ケースを条件分岐で書かないようにしましょう。

ヒント 2
C 言語では、文字型のデータはパソコン内部では下記表の数値（バイトコード）で管理されています。
また、文字と数値の演算や、文字同士の大小比較も可能です。
例：`printf("%c", 'N' + 5);`　→　「S」が出力される
例：`'n' < 'b'`　→　比較結果は `false`

65 A	66 B	67 C	68 D	69 E	70 F	71 G	72 H	73 I	74 J	75 K	76 L	77 M	78 N	79 O	80 P	81 Q	82 R	83 S	84 T	85 U	86 V	87 W	88 X	89 Y	90 Z

97 a	98 b	99 c	100 d	101 e	102 f	103 g	104 h	105 i	106 j	107 k	108 l	109 m	110 n	111 o	112 p	113 q	114 r	115 s	116 t	117 u	118 v	119 w	120 x	121 y	122 z

ヒント 3
Java では、**文字列.toUpperCase()**で文字列を全て大文字化、**文字列.toLowerCase()**で文字列を全て小文字化した結果を返します。
例：`System.out.println("n".toUpperCase())`　→　「N」が出力される
例：`System.out.println("S".toLowerCase())`　→　「s」が出力される
また、文字型変数を文字列型変数への変換は `Character.toString(文字型変数)`を使います。

開始時刻　　時　　分　　終了時刻　　時　　分　　所要時間　　時間　　分

演習5　16 進数表記の 10 進数変換　（30 分）

ファイル名：C 言語　…　enshu05.c　　~~Java　…　Enshu05.java~~

この問題は C 言語のみの演習です。
文字型の変数 beforeHex に初期化のタイミングで任意の'0'～'9'、'a'～'f'のいずれか 1 文字を設定し、入力された文字を 16 進数の 1 文字として考え、10 進数の int 型の値に変換して出力しましょう。
なお、変換処理は hex2dec 関数として記述してください。
上記指定以外の入力は無いものとします。
大文字の'A'～'F'も入力されないものとします。
ただし、C 言語の場合は任意の 1 文字を入力させるよう実装しても良いものとします。
また、'0'のバイトコードは 48、'a'のバイトコードは 97 です。

beforeHex の値	出力値
0	0
1	1
:	:
9	9
a	10
b	11
:	:
f	15

ヒント 1
この問題は文字列をバイトコードとして扱って演算ができる C 言語だけの演習です。

ヒント 2
プログラムは比較回数が多いと処理にかかる時間が長くなりやすいです。
できるだけ比較をしない記述を考えてみましょう。

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

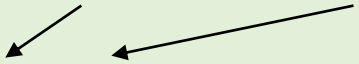
演習6　ユークリッドの互除法による最大公約数の算出　（30 分）

ファイル名：C 言語　…　enshu06.c　　Java　…　Enshu06.java

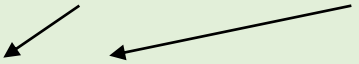
任意の整数 2 つに関して、最大公約数を求めるプログラムを作りましょう。
最大公約数を求めるアルゴリズムとして「ユークリッドの互除法」を使ってください。
ただし、C 言語の場合は任意の 2 数を入力させるよう実装しても良いものとします。

ヒント 1
※ユークリッドの互除法
例）96 と 60 の場合

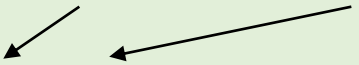
96　÷　60　＝　1　あまり　36　←あまりが 0 でないときは割る数をあまりで割る



60　÷　36　＝　1　あまり　24　←あまりが 0 になるまで繰り返す



36　÷　24　＝　1　あまり　12　←あまりが 0 になるまで繰り返す



24　÷　**12**　＝　2　あまり　0　←あまりが 0 になったタイミングの割る数が最大公約数

96 と 60 の最大公約数は 12

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習7　複数条件の集約　（40 分）

ファイル名：C 言語　…　enshu07.c　　Java　…　Enshu07.java

整数型の変数 target に初期化のタイミングで任意の整数を設定し、下記条件のいずれかを満たしている場合は「◎」、満たしていない場合は「×」を出力しましょう。
条件部については簡略化した記述ができるようであれば短い記述で比較回数が少なくできれば好ましい。ただし、C 言語の場合は任意の数を入力させるよう実装しても良いものとします。

条件

A：2 の倍数であり、3 の倍数であり、5 の倍数ではなく、7 の倍数ではない
B：2 の倍数であり、3 の倍数であり、5 の倍数ではなく、7 の倍数である
C：2 の倍数であり、3 の倍数であり、5 の倍数であり、7 倍数である
D：2 の倍数ではなく、3 の倍数ではなく、5 の倍数ではなく、7 の倍数ではない
E：2 の倍数ではなく、3 の倍数であり、5 の倍数であり、7 の倍数ではない
F：2 の倍数ではなく、3 の倍数であり、5 の倍数ではなく、7 の倍数ではない
G：2 の倍数であり、3 の倍数ではなく、5 の倍数ではなく、7 の倍数ではない
H：2 の倍数ではなく、3 の倍数であり、5 の倍数であり、7 の倍数である
I：2 の倍数であり、3 の倍数ではなく、5 の倍数ではなく、7 の倍数である

ヒント 1
条件 A と条件 B をよく見比べてみましょう。最後の条件「7 の倍数かどうか」はどちらでも構わないことに気が付きましたか？このようにして条件の記述を簡略化していきます。

ヒント 2
下記のカルノー図で各条件を書き出して、2^n 個（2 個・4 個・8 個・16 個…）の長方形や正方形で囲んで(反対側に回っても OK、他のグループと重なっても OK、できるだけ多くを含むグループになるように、グループ数は少なくなるように)グループ化し、条件の簡略化を考えてみましょう。

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習8　期間重複チェック　（30 分）

ファイル名：C 言語　…　enshu08.c　　Java　…　Enshu08.java

とある月(1～31 日)の中で、セール期間の開始日と終了日を設定しようと思います。
10 日～15 日は改装工事の期間なので、セール期間と重なる日がないかを確認・表示させましょう。
ただし、ループ文は使わず、セール期間との重複日があるかを確認する if 文は 1 つにまとめてください。
複数条件を OR、AND でつなげるのは構いません。
ただし、C 言語の場合は開始日・終了日に任意の 2 数を入力させるよう実装しても良いものとします。
また、セール期間の終了日は必ず開始日以降の日付であるものとします。

ヒント 1
セール期間の開始日と工事期間の開始日、セール期間の開始日と工事期間の終了日、セール期間の終了日と工事期間の開始日、セール期間の終了日と工事期間の終了日、それぞれの比較結果で演習 7 と同じようにカルノー図を書いてみましょう。(同じ日のケースは後回しで考えた方が答えにたどり着きやすいかもしれません)

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習9　素因数分解の繰り返し　（40 分）

ファイル名：C 言語　…　enshu09.c　　Java　…　Enshu09.java

4000 から 5000 までの整数のうち、17 の倍数について素因数分解の結果をすべて表示させましょう。
掛け算のみでの表現（例 1）でも構いませんが、指数表現（例 2）が出来ればなお良いです。

例 1　`4624 = 2 * 2 * 2 * 2 * 17 * 17`

例 2　`4624 = 2 ^ 4 * 17 ^ 2`

ヒント 1

記述すべきことがたくさん出てきましたね。どれをどの順で記述すれば良いのかを考えましょう。

・4000 から 5000 の範囲で繰り返す
・17 の倍数かどうかを判断する
・素因数分解して、表示する

ヒント 2

素因数分解は、「割ってあまりが出ないなら何度も割り算する」を 2 から順に全て割り切ってしまうまで繰り返します。

ヒント 3

指数表現をするときは「何回割り切れたか」の回数を保持しておくと作りやすくなります。

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習10 整数値を漢数字表記に変換　（30 分）

ファイル名：C 言語　…　enshu10.c　　Java　…　Enshu10.java

整数型の変数 arabia に初期化のタイミングで 0～20 億の範囲で任意の整数を設定し、その値を漢数字で出力させましょう。
下記例 2 のようなケースで「**四億零千零百零十零万零千零百零十零**」のような一般的でない表記は避けられれば好ましいです。
ただし、C 言語の場合は任意の数を入力させるよう実装しても良いものとします。

例 1　入力：1230456708　出力：**十二億三千四十五万六千七百八**
例 2　入力：400000000　出力：**四億**

ヒント 1
大きい桁から順に 1 桁ずつ取り出して、漢数字に変換していきましょう。
1 桁ずつ取り出す処理は、「**%**」（剰余の算出）を有効に使いましょう。

ヒント 2
「1500001004」のような入力の時に正常な結果が出力されるか確認してみましょう。

開始時刻＿＿時＿＿分　終了時刻＿＿時＿＿分　所要時間＿＿時間＿＿分

演習11 2進数表記の10進数変換　（30分）

ファイル名：C言語　…　enshu11.c　　Java　…　Enshu11.java

プログラム内で「**int binVal = 11001010;**」と記述すると、見た目は2進数表記のようですが、このままだと10進数の整数値「**千百万千十**」となってしまいます。下記ソースコードのコメント部に処理を追加し、10進数に変換した値(binValが11001010のときは、202)が出力できるようにしましょう。

C言語　※include文は省略しています。

```
int main(void) {
        int binVal = 11001010;
        int decVal;
        // ***** ここに10進数変換処理を記述 *****
        printf("%dは10進数に変換すると%dです。¥n", binVal, decVal);
}
```

Java　※パッケージやclass記述は省略しています。

```
public static void main(String[] args) {
        int binVal = 11001010;
        int decVal;
        // ***** ここに10進数変換処理を記述 *****
        System.out.println(binVal + "は10進数に変換すると" + decVal + "です。");
}
```

ヒント1
2進数での **11001010** は10進数に直す際の計算は、各桁の値を取り出して
1 * 128 + **1** * 64 + **0** * 32 + **0** * 16 + **1** * 8 + **0** * 4 + **1** * 2 + **0** * 1
で表せます。

ヒント2
ヒント1に出てくる128や64といった値はそれぞれ、2^7や、2^6といった計算で表せます。

ヒント3
a^bのような累乗は各言語で下記のように記述します。
C言語：pow(a, b)　ただし、事前に#include <math.h>の記述が必要です。
Java：Math.pow(a, b)

開始時刻＿＿時＿＿分　終了時刻＿＿時＿＿分　所要時間＿＿時間＿＿分

演習12 シンプルソート　（30 分）

ファイル名：C 言語　…　enshu12.c　　Java　…　Enshu12.java

整数型の配列 rndArray の内容を**シンプルソート**で昇順に並べ出力しましょう。
ただし、rndArray の中身については、乱数を使用しても、C 言語の場合は任意の数を入力させるよう実装しても良いものとします。

ヒント１
シンプルソートのアルゴリズムは、「一番小さいものを探し、結果の配列にセットしていく。」です。
人間が一番理解しやすい単純(simple)な考え方ですが、他の全ての値との比較を繰り返す処理は、比較する回数が多く、ソートのアルゴリズムとしては非効率的です。

アルゴリズム例
未整列配列：２５３１６　→　最小値：１　→　結果配列：１
未整列配列：２５３６　→　最小値：２　→　結果配列：１２
未整列配列：５３６　→　最小値：３　→　結果配列：１２３
未整列配列：５６　→　最小値：５　→　結果配列：１２３５
未整列配列：６　→　最小値：６　→　結果配列：１２３５６

開始時刻＿＿時＿＿分　終了時刻＿＿時＿＿分　所要時間＿＿時間＿＿分

演習13 バブルソート　（30 分）

ファイル名：C 言語　…　enshu13.c　　Java　…　Enshu13.java

整数型の配列 rndArray の内容を**バブルソート**で昇順に並べ出力しましょう。
ただし、rndArray の中身については、乱数を使用しても、C 言語の場合は任意の数を入力させるよう実装しても良いものとします。

ヒント１
バブルソートのアルゴリズムは、「隣り合う 2 数を前から順に比較して、2 数が小さい数、大きい数の順になるよう入れ替える処理を繰り返す。」です。
最大値が浮かびあがってくる様が泡(bubble)のようであることからこの名称がついています。

アルゴリズム例
2 5 3 1 6　　a b(下線)…a と b を比較、b の方が小さいなら入れ替える
2 5 3 1 6
2 3 5 1 6
2 3 1 5 6
2 3 1 5 **6**　　**a(太字)**…順番が確定した値
2 3 1 5 **6**
2 1 3 5 **6**
2 1 3 **5 6**
1 2 3 **5 6**
1 2 **3 5 6**
1 2 3 5 6

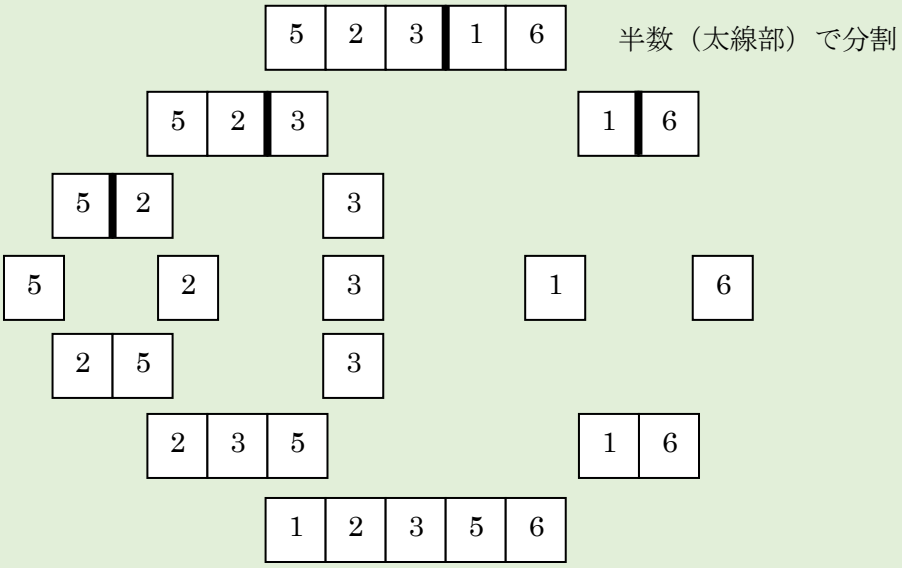
開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習14 マージソート　（60 分）

ファイル名：C 言語　…　enshu14.c　　Java　…　Enshu14.java

整数型の配列 rndArray の内容を**マージソート**で昇順に並べ出力しましょう。
ただし、rndArray の中身については、乱数を使用しても、C 言語の場合は任意の数を入力させるよう実装しても良いものとします。

ヒント 1
マージソートのアルゴリズムは、「配列を 2 分割していき、その後くっつける(merge)際に 2 つの配列の先頭同士を比べ、小さいものから取り出す。」です。
再帰呼び出しの考え方が必要になり、多少複雑になりますが、くっつける為の比較の際、2 つの配列が昇順（先頭要素が常に最小値同士）になるため、比較回数が減らせます。



ヒント 2
再帰呼び出しとは、下記のように、関数の中でその関数自身をさらに呼び出す処理です。

```
int kaijou(int n) {
	if (n > 0) kaijou(n - 1);
}
```

ヒント 3
配列のコピーは C 言語では **memcpy(コピー先配列開始ポインタ，コピー元配列開始ポインタ，コピーバイト数)**、Java では **System.arraycopy(コピー元配列，コピー元開始インデックス，コピー先配列，コピー先開始インデックス，コピー要素数)**で実装できます。

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習15 ヒープソート　（60 分）

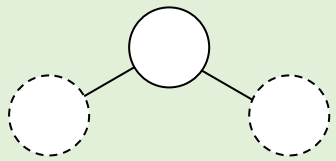
ファイル名：C 言語　…　enshu15.c　　Java　…　Enshu15.java

整数型の配列 rndArray の内容を**ヒープソート**で昇順に並べ出力しましょう。
ただし、rndArray の中身については、乱数を使用しても、C 言語の場合は任意の数を入力させるよう実装しても良いものとします。

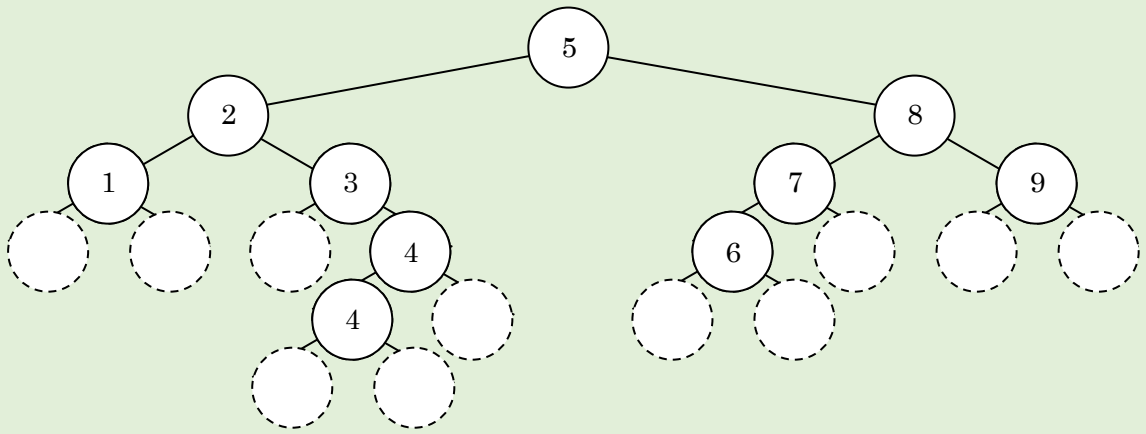
ヒント 1
ヒープソートのアルゴリズムは、「データを 2 分木構造の繰り返しで表現して、左の枝には自分の値以下のものを、右の枝には自分より大きい値のものをつなげる。これを繰り返す。」です。
再帰呼び出しの考え方と、構造体またはクラスの考え方が必要になります。
形状が山積み(heap)の形になることから、この名称がついています。

ヒント 2
2 分木構造は C 言語なら構造体、Java ならクラスで実装します。

基本の 2 分木の構造



ヒント 3
入力が「5 8 2 3 4 7 1 9 4 6」の場合、下記のような木構造になります。（入力値の順番によって出来上がる木構造の形は異なります。）



開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習16 横棒グラフの作成　（20 分）

ファイル名：C 言語　…　enshu16.c　　Java　…　Enshu16.java

次のデータを int 型の配列 pointArray として保持し、出力例に従って横棒グラフを表示させましょう。
テストの最高得点は 100 点とします。
ただし、C 言語の場合は任意の数を入力させるよう実装しても良いものとします。
データ：　1 人目…68 点　2 人目…9 点　3 人目…100 点　4 人目…81 点
出力例：

```
1 :******
2 :
3 :**********
4 :********
```

ヒント 1
点数を 10 点区切りで確認して、「10 点を超えていたら'*'を出力。20 点を超えていたら'*'をもう一つ出力、30 点を超えていたら…」と繰り返します。

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習17 縦棒グラフの作成　（30 分）

ファイル名：C 言語　…　enshu17.c　　Java　…　Enshu17.java

次のデータを int 型の配列 pointArray として保持し、出力例に従って縦棒グラフを表示させましょう。
テストの最高得点は 100 点とします。
ただし、C 言語の場合は任意の数を入力させるよう実装しても良いものとします。
データ：　1 人目…68 点　2 人目…9 点　3 人目…100 点　4 人目…81 点
出力例：

```
    *
    *
    * *
    * *
*   * *
*   * *
*   * *
*   * *
*   * *
*   * *
====
1 2 3 4
```

ヒント 1
C 言語や Java では縦方向の出力はできず、横方向 1 行ずつの出力になります。
出力例の各行に下線を引いてみましょう。
1 人目が各行で'*'が出力されるのはどのような時かを考えてみましょう。

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習18 ファイル内の単語頻度集計　（120 分）

ファイル名：C 言語　…　enshu18.c　　Java　…　Enshu18.java

下記仕様のプログラムを作成しましょう。

・コマンドライン引数でファイルを指定します。(以下、「対象ファイル」とします)
・対象ファイルが無い・読み込めない場合は、「**ファイルが読み込めません**」と表示し、終了します。
・対象ファイルはプログラムのソースコード(テキストデータ)が望ましいです。
・対象ファイルをタブ文字およびスペース文字で区切り、単語に分けます。
・対象ファイル内で記述されていた単語(以下、「出現単語」とする)の出現数をカウントします。
・出現単語を辞書順(**記号、0～9、a～z**　の順)で、出現数とともに一覧出力しましょう。
・大文字・小文字は区別せず、同一の単語として扱います。
・文字列("～"で囲まれている)や文字('～'で囲まれている)は出現単語として扱いません。
・一行コメント(//～　の形式)やブロックコメント(/*～*/　の形式)は出現単語として扱いません。
・記号（{、}、(、)、=、+など）は出現単語の一部として扱いません。
　ただし、#、$、_については出現単語の一部として扱ってください。
　数字については、変数名の一部として使われている文字は出現単語の一部として扱ってください。

入力例：

```
int getLastKeta(int param) {
        // 保持用の変数
        int result1 = 0;

/* result1 = param - (param / 10) * 10; */
        result1 = param % 10;
        return result1;
}
```

出力例：

```
getLastKeta    1
int     3
param   2
result1 3
return  1
```

ヒント１
全機能を一気に完成させようとせず、一つずつ動作確認しながら実装しましょう。

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習19 15 進数の 10 進数変換と 100 乗　（120 分）

ファイル名：C 言語　…　enshu19.c　　Java　…　Enshu19.java

15 進数(0～9,A～E)で表現された 4 桁のデータ(0000～EEEE、10 進数に直すと 0～50624)を 10 進数に変換し、100 乗した結果を出力させましょう。
ただし、C 言語の場合は任意の数を入力させるよう実装しても良いものとします。
15 進数の表記は大文字・小文字のどちらでも処理できるものとします。

ヒント 1
100 乗した値は非常に大きな桁になります。int 型はもちろんのこと、long 型でも余裕でオーバーフローします。変数 1 つだけで全桁を保持するのは無理なので、配列を使うなどして分割して保持するようにしましょう。
2 を 100 乗しただけでも、
2^{100} = 1,267,650,600,228,229,401,496,703,205,376　と、31 桁にもなります。
なお、最大値 50624 を 100 乗すると、
272675267686374925574231851062393542579205012793661157498857923973908640477794207712986214799712270894610762854045090258071175822777157229146227091549031311295358720926763689787064418482651245516821959756010510361741961418300332506008740510020352308689776273369542435107300544339938156069533063371101246467433227669358415218281021798662768422900755531872552213342530520448293662010000006826913904703795097578602014580074832242042498866289271026338730748670027560571109376
と、最大で 471 桁にまでなります。

ヒント 2
100 乗を一気に計算しようとせずに、同じ数を 100 回かけ算するよう実装しましょう。

ヒント 3
まずは小さいデータで、累乗回数も少ない状態で動作確認してからにしましょう。

開始時刻＿＿＿時＿＿＿分　終了時刻＿＿＿時＿＿＿分　所要時間＿＿＿時間＿＿＿分

演習20 双子の素数　（90 分）

ファイル名：C 言語　…　enshu20.c　　Java　…　Enshu20.java

10000 までの素数の中で双子の素数(3 と 5、11 と 13 のように素数同士の差が 2 のペア)を探し、出力しましょう。

ヒント 1
特定の数が素数かどうかは、その数より小さい数で割り算してみて、割り切れることがあるかを確認します。

ヒント 2
より効率的なソースを書きたい方は、素数を抽出するアルゴリズムである「エラトステネスの篩(ふるい)」を取り入れてみましょう。

エラトステネスの篩のアルゴリズム
・配列に 2 から調べたい最大値までの整数値を昇順で入れます。
・配列の先頭から順に、対象外となっていない値は素数として保持しておき、その倍数を全て素数の対象外とします。
・この作業を繰り返します。
・対象外にならなかった値を全て素数として扱います。