問３　多倍長整数の演算に関する次の記述を読んで，設問に答えよ。

コンピュータが一度に処理できる整数の最大桁には，CPU が一度に扱える情報量に依存した限界がある。一度に扱える桁数を超える演算を行う一つの方法として，10 を基数とした多倍長整数（以下，多倍長整数という）を用いる方法がある。

〔多倍長整数の加減算〕

多倍長整数の演算では，整数の桁ごとの値を，1 の位から順に 1 次元配列に格納して管理する。例えば整数 123 は，要素数が 3 の配列に{3, 2, 1}を格納して表現する。

多倍長整数の加算は，“桁ごとの加算”の後，“繰り上がり”を処理することで行う。456＋789 を計算した例を図 1 に示す。

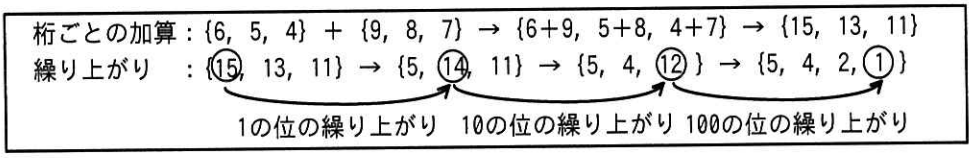


図 1　456＋789 を計算した例

“桁ごとの加算”を行うと，配列の内容は{15, 13, 11}となる。1 の位は 15 になるが，15 は 10×1＋5 なので，10 の位である 13 に 1 を繰り上げて{5, 14, 11}とする。これを最上位まで繰り返す。最上位で繰り上がりが発生する場合は，配列の要素数を増やして対応する。減算も同様に“桁ごとの減算”と“繰り下がり”との処理で計算できる。

〔多倍長整数の乗算〕

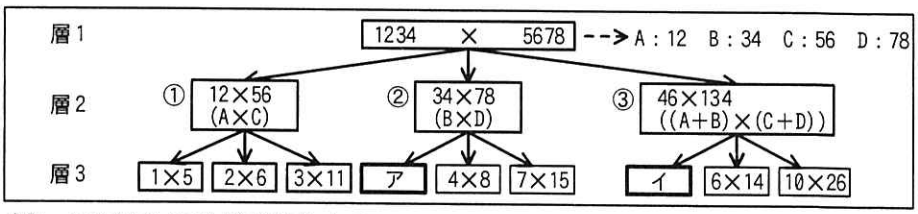
多倍長整数の乗算については，計算量を削減するアルゴリズムが考案されており，その中の一つにカラツバ法がある。ここでは，桁数が 2 のべき乗で，同じ桁数をもった正の整数同士の乗算について，カラツバ法を適用した計算を行うことを考える。桁数が 2 のべき乗でない整数や，桁数が異なる整数同士の乗算を扱う場合は，上位の桁を 0 で埋めて処理する。例えば，123×4 は 0123×0004 として扱う。

〔ツリー構造の構築〕

カラツバ法を適用した乗算のアルゴリズムは，計算のためのツリー構造（以下，ツリーという）を作る処理と，ツリーを用いて演算をする処理から成る。ツリーは，多倍長整数の乗算の式を一つのノードとし，一つのノードは 3 個の子ノードをもつ。

M 桁×M 桁の乗算の式について，乗算記号の左右にある値を，それぞれ M/2 桁ずつに分けて A，B，C，D の四つの多倍長整数を作る。これらの整数を使って，①A×C，②B×D，③(A＋B)×(C＋D)の 3 個の子ノードを作り，M/2 桁×M/2 桁の乗算を行う層を作る。(A＋B)，(C＋D)は多倍長整数の加算の結果であるが，ここでは“桁ごとの加算”だけを行い，“繰り上がり”の処理はツリーを用いて行う演算の最後でまとめて行う。生成した子ノードについても同じ手順を繰り返し，1 桁×1 桁の乗算を行う最下層のノードまで展開する。

1234×5678 についてのツリーを図 2 に示す。図 2 の層 2 の場合，①は 12×56，②は 34×78，③は 46×134 となる。③の(C＋D)は，“桁ごとの加算”だけの処理を行うと，10 の位が 5＋7＝12，1 の位が 6＋8＝14 となるので，12×10＋14＝134 となる。



注記　この例では層 3 が最下層となる。

図 2　1234×5678 についてのツリー

〔ツリーを用いた演算〕

ツリーの最下層のノードは，整数の乗算だけで計算できる。最下層以外の層は，子ノードの計算結果を使って，次の式で計算できることが分かっている。ここで，α，β，γは，それぞれ子ノード①，②，③の乗算の計算結果を，K は対象のノードの桁数を表す。

$$\alpha \times 10^{K} + (\gamma - \alpha - \beta) \times 10^{K/2} + \beta \quad \cdots\cdots(1)$$

図 2 のルートノードの場合，K＝4，α＝672，β＝2652，γ＝6164 なので，計算結果は次のとおりとなる。

$$672 \times 10000 + (6164 - 672 - 2652) \times 100 + 2652 = 7006652$$

〔多倍長整数の乗算のプログラム〕

桁数が 2 のべき乗の多倍長整数 val1，val2 の乗算を行うプログラムを作成した。

プログラム中で利用する多倍長整数と，ツリーのノードは構造体で取り扱う。構造体の型と要素を表 1 に示す。構造体の各要素には，構造体の変数名.要素名でアクセスできる。また，配列の添字は 1 から始まる。

表 1　構造体の型と要素

構造体の型	要素名	要素の型	内容
多倍長整数	N	整数	多倍長整数の桁数
	values	整数の配列	桁ごとの値を管理する 1 次元配列。1 の位の値から順に値を格納する。配列の要素は，必要な桁を全て格納するのに十分な数が確保されているものとする。
ノード	N	整数	ノードが取り扱う多倍長整数の桁数。図 2 の 1234×5678 のノードの場合は 4 である。
	val1	多倍長整数	乗算記号の左側の値
	val2	多倍長整数	乗算記号の右側の値
	result	多倍長整数	乗算の計算結果

多倍長整数の操作を行う関数を表 2 に，プログラムで使用する主な変数，配列及び関数を表 3 に，与えられた二つの多倍長整数からツリーを構築するプログラムを図 3 に，そのツリーを用いて演算を行うプログラムを図 4 に，それぞれ示す。表 2，表 3 中の p，q，v1，v2 の型は多倍長整数である。また，図 3，図 4 中の変数は全て大域変数である。

表 2　多倍長整数の操作を行う関数

名称	型	内容
add(p, q)	多倍長整数	p と q について，“桁ごとの加算” を行う。
carry(p)	多倍長整数	p について “繰り上がり”・“繰り下がり” の処理を行う。
left(p, k)	多倍長整数	p について，values の添字が大きい方の k 個の要素を返す。 p の values が{4, 3, 2, 1}，k が 2 であれば，values が{2, 1}の多倍長整数を返す。
right(p, k)	多倍長整数	p について，values の添字が小さい方の k 個の要素を返す。 p の values が{4, 3, 2, 1}，k が 2 であれば，values が{4, 3}の多倍長整数を返す。
lradd(p, k)	多倍長整数	add(left(p, k), right(p, k))の結果を返す。
shift(p, k)	多倍長整数	p を 10^k 倍する。
sub(p, q)	多倍長整数	p と q について，“桁ごとの減算” を行い p−q を返す。

表3　使用する主な変数，配列及び関数

名称	種類	型	内容
elements[]	配列	ノード	ツリーのノードを管理する配列。ルートノードを先頭に，各層の左側のノードから順に要素を格納する。図 2 の場合は，{1234×5678，12×56，34×78，46×134，1×5，2×6，…}の順で格納する。
layer_top[]	配列	整数	ルートノードから順に，各層の左端のノードの，elements 配列上での添字の値を格納する。図 2 の場合は 1234×5678，12×56，1×5 の添字に対応する{1，2，5}が入る。
mod(m, k)	関数	整数	m を k で割った剰余を整数で返す。
new_elem(k,v1,v2)	関数	ノード	取り扱う多倍長整数の桁数が k で，v1×v2 の乗算を表すノード構造体を新規に一つ作成して返す。
pow(m, k)	関数	整数	m の k 乗を整数で返す。k が 0 の場合は 1 を返す。
t_depth	変数	整数	ツリーの層の数。図 2 の場合は 3 である。
val1, val2	変数	多倍長整数	乗算する対象の二つの値。図 2 の場合，ルートノードの二つの値で，val1 は 1234，val2 は 5678 である。
answer	変数	多倍長整数	乗算の計算結果を格納する変数

```
// ツリーの各層の，elements配列上での先頭インデックスを算出する
layer_top[1] ← 1                                    // ルートノードは先頭なので1を入れる
for (iを1からt_depth − 1まで1ずつ増やす)
  layer_top[i + 1] ← layer_top[i] + [ ウ ]
endfor

// ツリーを構築する
elements[1] ← new_elem(val1.N, val1, val2)      // ルートノードを用意。桁数はval1の桁数を使う
for (dpを1からt_depth − 1まで1ずつ増やす)       // ルートノードの層から，最下層以外の層を順に処理
  for (iを1からpow(3, dp − 1)まで1ずつ増やす)   // 親ノードになる層の要素数だけ繰り返す
    pe ← elements[layer_top[dp] + (i − 1)]      // 親ノードの要素を取得
    cn ← pe.N / 2                               // 子ノードの桁数を算出
    tidx ← layer_top[dp + 1] + [ エ ]           // 子ノード①へのインデックス
    elements[tidx     ] ← new_elem(cn, left([ オ ], cn), left([ カ ], cn))
    elements[tidx + 1] ← new_elem(cn, right([ オ ], cn), right([ カ ], cn))
    elements[tidx + 2] ← new_elem(cn, lradd([ オ ], cn), lradd([ カ ], cn))
  endfor
endfor
```

図3　与えられた二つの多倍長整数からツリーを構築するプログラム

```
// 最下層の計算
for (iを1からpow(3, t_depth − 1)まで1ずつ増やす)          // 最下層の要素数は3のt_depth−1乗個
  el ← elements[layer_top[t_depth] + (i − 1)]            // 最下層のノード
  mul ← el.val1.values[1] * el.val2.values[1]            // 最下層の乗算
  el.result.N ← 2                                        // 計算結果は2桁の多倍長整数
  el.result.values[1] ← [ キ ]                           // 1の位
  el.result.values[2] ← mul / 10                         // 10の位
endfor

// 最下層以外の計算
for (dpをt_depth − 1から1まで1ずつ減らす)                  // 最下層より一つ上の層から順に処理
  for (iを1からpow(3, dp − 1)まで1ずつ増やす)              // 各層の要素数だけ繰り返す
    el ← elements[layer_top[dp] + (i − 1)]               // 計算対象のノード
    cidx ← layer_top[dp + 1] + [ エ ]                    // 子ノード①へのインデックス
    s1 ← sub( [ ク ].result, [ ケ ].result )
    s2 ← sub(s1, elements[cidx + 1].result)              // γ−α−βを計算
    p1 ← shift(elements[cidx].result, el.N)              // α×10^K を計算
    p2 ← shift(s2, el.N / 2)                             // (γ−α−β)×10^(K/2) を計算
    p3 ← elements[cidx + 1].result                       // βを計算
    el.result ← add(add(p1, p2), p3)
  endfor
endfor

// 繰り上がり処理
answer ← carry(elements[1].result)                       // 計算結果をanswerに格納
```

注記　図4中の［エ］には，図3中の［エ］と同じ字句が入る。

図4　ツリーを用いて演算を行うプログラム

設問1　図2中の［ア］，［イ］に入れる適切な字句を答えよ。

設問2　図2中の層2にある46×134のノードについて，本文中の式(1)の数式は具体的にどのような計算式になるか。次の式の①～④に入れる適切な整数を答えよ。

(①)×100 + ((②)−(③)−84)×10 + (④)

設問3　図3中の［ウ］～［カ］に入れる適切な字句を答えよ。

設問4　図4中の［キ］～［ケ］に入れる適切な字句を答えよ。

設問5　N桁同士の乗算をする場合，多倍長整数の構造体において，配列 values に必要な最大の要素数は幾つか。Nを用いて答えよ。