＜擬似言語演習問題①　1～10 の総和＞

問　変数 SU の値を 1～10 又は 10～1 まで変化させ，その和を ANS に求める。

［プログラム 1］

○ プログラム名：　1～10 の総和 1

○ 整数型：　SU，ANS

・ ANS←0

・ SU←0

SU　＜　10　（別解：SU　≦　9）

・ SU←SU＋1　SUの初期値は0なので、＋1をしてから和を計算する

・ ANS←ANS＋SU

［プログラム 2］

○ プログラム名：　1～10 の総和 2

○ 整数型：　SU，ANS

・ ANS←0

・ SU←1

和の計算(ANS←ANS+SU)を先にしているので、
ＳＵの値が１１になった時が繰り返しの終了となる

SU　＜　11　（別解：SU　≦　10）

・ ANS←ANS＋SU　SUの初期値が1なので、和を先に計算する

・ SU←SU＋1

［プログラム 3］

○ プログラム名：　10～1 の総和 3

○ 整数型：　SU，ANS

・ ANS←0

・ SU←10

SU　＞　0　（別解：SU　≧　1）

・ ANS←ANS＋SU　SUの初期値が１０なので、和を先に計算する

・ SU←SU－1

［プログラム 4］

○ プログラム名：　10～1 の総和 4

○ 整数型：　SU，ANS

・ ANS←0

・ SU←11

SU　＞　1　（別解：SU　≧　2）

・ SU←SU－1　SUの初期値が１１なので、－１をしてから
和を計算する

・ ANS←ANS＋SU

＜擬似言語演習問題②　加算の繰り返しによる乗算＞

問　ANS←m×n を，加算の繰り返しによって行う（m を n 回加算する）。

［プログラム 1］

○ プログラム名：　加算乗算 1

○ 整数型：　ANS，m，n

・ m←被乗数

・ n←乗数　nが繰り返しの回数

・ ANS←0

■ n　＞　0

・ ANS←ANS＋m

・ n←n－1

■

［プログラム 2］

○ プログラム名：　加算乗算 2

○ 整数型：　ANS，m，n，CNT

CNTを繰り返し処理の判定に利用

・ m←被乗数

・ n←乗数

・ ANS←0

・ CNT←0

■ CNT　＜　n

・ ANS←ANS＋m

・ CNT←CNT＋1

■

別解

・CNT←n

■CNT ＞ 0

｜・ANS←ANS＋m

｜・CNT←CNT－1

■

［プログラム 3］

○ プログラム名：　加算乗算 3　＜加算回数を考慮し，効率化を図る＞

○ 整数型：　ANS，m，n，WK｛WK は作業領域｝

・ m←被乗数

・ n←乗数

・ ANS←0

m　＜　n　　繰り返し回数を少なくするため、mとnを比較して小さい値をnにする

・ WK←m

・ m←n

・ n←WK

WKを利用して
mとnを交換する

n　＞　0

・ n←n－1

・ ANS←ANS＋m

プログラム1の処理と同じ

＜擬似言語演習問題③　減算の繰り返しによる除算＞

問　m÷n（m≧n）を，減算の繰り返しによって行い，商を SYO，余りを AMR に求める。

考え方
被除数÷除数＝商･･･余り

[例]　13÷4＝3･･･1
13－4＝9
9－4＝5
5－4＝1　(1は4より小さいので終了)
3回引けたので商は3
最後に残った値の1が余り

［プログラム 1］

○ プログラム名：　減算除算 1
○ 整数型：　m，n，SYO，AMR
- m←被除数
- n←除数
- SYO←0
- m　≧　n
  - m←m－n
  - SYO←SYO＋1　引くことができた回数(SYO)を＋1する
- AMR←m

［プログラム 2］

○ プログラム名：　減算除算 2
○ 整数型：　m，n，SYO，AMR
- m←被除数
- n←除数
- SYO←0
- AMR←m
- AMR　≧　n
  - AMR←AMR－n
  - SYO←SYO＋1

考え方はプログラム1と同じ
mの代わりにAMRを使用して計算

［プログラム 3］

○ プログラム名：　減算除算 3　＜加算の繰り返しで除算を実現する＞
○ 整数型：　m，n，SYO，AMR，WK｛WK は作業領域｝
- m←被除数
- n←除数
- SYO←0
- WK←n
- m　≧　WK
  - WK←WK＋n
  - SYO←SYO＋1
- WK←WK－n
- AMR←m－WK

考え方

[例]　13÷4＝3･･･1
4＋4＝8
8＋4＝12
12＋4＝16　(被除数の13を超えたので終了)
3回足すことができたので商は3
余りは終了前の和12を被除数13から引いて1

＜擬似言語演習問題④　配列要素の最大値及び最小値＞

問　要素数 100 の一次元配列 TBL（指標：i）に値が格納されているとき，その要素の最大値を MAX に、最小値を MIN に格納する。指標の値は 0 からとする。

［プログラム］

○ プログラム名：　最大値最小値
○ 整数型：　i，MAX，MIN
○ 整数型：　TBL[100]｛配列要素はあらかじめ格納されている｝

・ i←0
・ MAX←TBL[i]
・ MIN←MAX

ポイント
← 配列要素が全てマイナスの場合を考慮して、MAXとMINの初期値は0ではなく、最初の要素を設定する

■ i ＜ 99

i＜100だと、iが99の時に＋1をすると指標の値が100になってしまう

・ i←i＋1
MAX ＜ TBL[i]
・ MAX←TBL[i]

MIN ＞ TBL[i]
・ MIN←TBL[i]

■

＜擬似言語演習問題⑤　配列要素の平均値＞

問　要素数 100 の一次元配列 TBL（指標：i）に値が格納されているとき，その要素が 0 を超えるものについて件数を CNT に数えながら SUM に合計し，平均を AVG に求める。指標の値は 0 からとする。

［プログラム 1］

○ プログラム名：　配列要素平均 1
○ 整数型：　SUM，CNT，AVG
○ 整数型：　TBL[100]｛配列要素はあらかじめ格納されている｝

```
・ SUM←0
・ CNT←0        CNTの初期化
・ i←0
■ i ＜ 100
▲   TBL[i] ＞ 0
      ・ SUM←SUM＋TBL[i]        合計と件数の計算　順不同
      ・ CNT←CNT＋1
┼
▼
    ・ i←i＋1        ここで指標を＋1しているので、
                     繰り返し条件は i＜100
■
・ AVG←SUM／CNT
```

［プログラム 2］

○ プログラム名：　配列要素平均 2
○ 整数型：　SUM，CNT，AVG
○ 整数型：　TBL[100]｛配列要素はあらかじめ格納されている｝

```
・ SUM←0        SUMの初期化
・ CNT←0
・ i←99        指標の初期値を99としていることから、配列を後ろから見ていくことがわかる
■ i ＞ -1        指標の範囲は0～99なので、0の時も繰り返す
▲   TBL[i] ＞ 0
      ・ SUM←SUM＋TBL[i]
      ・ CNT←CNT＋1
┼
▼
    ・ i←i－1
■
・ AVG←SUM／CNT
```

＜擬似言語演習問題⑥　線形探索＞

問　要素数 N の一次元配列 TBL（指標：i）の各領域にアルファベットが 1 文字ずつ格納されている。配列を先頭から順に探索し，一致する文字があった場合，RST にその指標番号（複数ある場合は最初に見つかったもの）を，見つからなかった場合は 0 を格納してプログラムを終了する。指標の値は 0 からとする。

［プログラム］

○ プログラム名：　線形探索
○ 整数型：　i，N，RST｛N には要素数が格納されている｝
○ 整数型：　TBL[]｛各領域にはアルファベットが 1 文字ずつ格納されている｝
○ 文字型：　MJ｛探索するアルファベットが 1 文字格納されている｝

・ RST←0
・ i←0

i ＜ N AND MJ ≠ TBL[i]　…　※

・ i←i＋1

繰り返しが終了するのは
・文字が見つかったとき　（MJ＝TBL［i］）
・配列を最後までみたとき　（i＝N）
のどちらか　（OR）
従って、繰り返し条件は上記条件の逆になる

i ＜ N　配列の中にあった

・ RST←i

※　ショートサーキットルール

繰返しが複合条件のプログラミングでは，先に記述された条件が満たされた場合，残りの条件は無視される。

つまり、
MJ≠TBL［i］ AND i＜N となっており、配列の中に検索文字が存在しなかった場合は『MJ≠TBL［i］』の条件が常に満たされるので、『i＜N』の条件は無視される。従って要素数を超えて検索してしまうことになる。

＜擬似言語演習問題⑦　線形探索・番兵法＞

問　要素数 N の一次元配列 TBL（指標：i）の各領域にアルファベットが 1 文字ずつ格納されている。配列を先頭から順に探索し，一致する文字があった場合，RST にその指標番号（複数ある場合は最初に見つかったもの）を，見つからなかった場合は 0 を格納してプログラムを終了する。指標の値は 0 からとする。

番兵とは…検索値を配列の最後の要素の次に格納し、検索で必ず見つかるようにすることで繰り返し条件を単純に記述できるようにしたもの

［プログラム］

○ プログラム名：　線形探索・番兵法
○ 整数型：　i，N，RST｛N には要素数が格納されている｝
○ 整数型：　TBL[]｛各領域にはアルファベットが 1 文字ずつ格納されている｝
○ 文字型：　MJ｛探索するアルファベットが 1 文字格納されている｝

・ i←0
・ RST←0
・ TBL[N]←MJ　**番兵の設定**

■ MJ ≠ TBL[i]　**繰り返し条件が「見つからない間」だけで済む**
　・ i←i＋1
■

i ＜ N　**i＜Nは「配列の中にあった」とき**
　・ RST←i

＜擬似言語演習問題⑧　二分探索＞

問　要素数 N の一次元配列 TBL の各領域に数値が昇順に格納されている。SU に格納されている数値と同じものがあるかを二分探索によって探索し，一致する文字があった場合，RST にその指標番号（複数ある場合は最初に見つかったもの）を，見つからなかった場合は 0 を格納してプログラムを終了する。指標の値は 0 からとする。

二分探索は、検索する範囲の中位置の値と比較して、それよりも検索値が大きければ検索範囲を後ろ半分とし、検索値が小さければ検索範囲を前半分として検索回数を少なくして速度向上を目指す手法

［プログラム］

○ プログラム名：　二分探索
○ 整数型：　N，SW｛N：要素数，SW：一致したとき 1 を格納する｝
○ 整数型：　LO，HI，MID｛LO：下限値，HI：上限値，MID：中点｝
○ 整数型：　SU，RST｛SU：探索する値が格納されている｝
○ 整数型：　TBL[]｛要素は昇順に格納されている｝

・ LO←0
・ HI←N－1
・ SW←0

↓ 配列の中に検索値がない場合、検索範囲が狭まり『LO＞HI』になってしまう

■ SW ＝ 0 AND LO ≦ HI
　・ MID←INT（（LO＋HI）／2）　｛INT（ ）は小数点以下を切り捨てる関数｝
　　SU ＜ TBL[MID]　**検索値が配列の前半分だった**
　　・ HI←MID－1
　　SU ＞ TBL[MID]　**検索値が配列の後ろ半分だった**
　　・ LO←MID＋1
　　・ SW←1　**上記2つの条件に合わないときつまり、SU＝TBL[MID]のとき**
■

SW ＝ 1　**見つかったとき**
　・ RST←MID
　・ RST←0

＜擬似言語演習問題⑨　セレクションソート１＞

問　一次元配列 TBL に数値が格納されているとき，その要素を昇順に整列させる。

例　要素数が4のとき
①TBL[0]とTBL[1]を比較
②TBL[0]とTBL[2]を比較
③TBL[0]とTBL[3]を比較
④TBL[1]とTBL[2]を比較
⑤TBL[1]とTBL[3]を比較
⑥TBL[2]とTBL[3]を比較
↑iを使用 ↑jを使用

［プログラム１］

○ プログラム名：　セレクションソート１
○ 整数型：　N，i，j，WK｛WK は作業領域｝
○ 整数型：　TBL[]｛要素は格納されている｝

```
・N←要素数－1
・i←0
■ i ＜ N
  ・j←i＋1
  ■ j ≦ N
    ▲ TBL[i] ＞ TBL[j]
      ・WK←TBL[i]
      ・TBL[i]←TBL[j]
      ・TBL[j]←WK
    ▼
    ・j←j＋1
  ■
  ・i←i＋1
■
```

WKを使ってTBL[i]とTBL[j]を交換

［プログラム２］

○ プログラム名：　セレクションソート２
○ 整数型：　N，i，WK｛WK は作業領域｝
○ 整数型：　TBL[]｛要素は格納されている｝

```
・N←要素数－1
■ N ＞ 0
  ・i←0
  ■ i ＜ N
    ▲ TBL[i] ＞ TBL[N]
      ・WK←TBL[i]
      ・TBL[i]←TBL[N]
      ・TBL[N]←WK
    ▼
    ・i←i＋1
  ■
  ・N←N－1
■
```

別解

```
・N←要素数－1
■ N＞0
│・i←N－1
│■ i≧0
││▲ TBL[i]＞TBL[j]
│││ TBL[i]とTBL[j]の交換
││▼
││・i←i－1
│■
│・N←N－1
■
```