＜擬似言語演習問題⑩　セレクションソート２＞

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。但し，要素の交換回数がすくなくなるよう改良されたアルゴリズムである。

i番目から最後までの要素のうち、最小値が入っている指標をminに格納する。
i番目の要素とmin番目の要素を入れ替える。
iの数を＋1して上記を行う処理を繰り返す。

［プログラム］

○ プログラム名：　セレクションソート改

○ 整数型：　N，min，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

・ min←i　　min：最小の要素が格納されている指標値が入る

・ j←i＋1

■ j ≦ N　　この繰り返しで最小値を探す

▲ TBL[j] ＞ TBL[min]

・ min←j

・ j←j＋1

■

・ WK←TBL[i]

・ TBL[i]←TBL[min]　　交換処理（ここで最小値がi番目に入る）

・ TBL[min]←WK

・ i←i＋1

■

＜擬似言語演習問題⑪　バブルソート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。

［プログラム］

○ プログラム名：　バブルソート

○ 整数型：　N，SW，i，WK｛SW：スイッチ，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

・ SW←1　↓1回も交換を行わなかったときは整列済みなので処理を終了する

■ SW　＝　1　AND　N　＞　0

　・ SW←0　一旦、交換を行わなかった(SW←0)にしておく

　・ i←0

　■ i　＜　N

　　TBL[i]　＞　TBL[i＋1]

　　・ WK←TBL[i]

　　・ TBL[i]←TBL[i＋1]　交換処理

　　・ TBL[i＋1]←WK

　　・ SW←1　交換を行ったとき、SWを1にする

　　・ i←i＋1

　■

　・ N←N－1

■

例　要素数が4のとき
①TBL[0]とTBL[1]を比較
②TBL[1]とTBL[2]を比較
③TBL[2]とTBL[3]を比較
④TBL[0]とTBL[1]を比較
⑤TBL[1]とTBL[2]を比較
⑥TBL[0]とTBL[1]を比較
↑iを使用　↑i＋1を使用
比較回数はセレクションソートと同じだが、SWにより比較回数が減る場合がある

＜擬似言語演習問題⑫　バブルソート２＞

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。

［プログラム］

○ プログラム名：　バブルソート２

○ 整数型：　N，SW，i，j，WK｛SW：スイッチ，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
・ j←0
・ SW←1
■ SW ＝ 1 AND j ＜ N
  ・ SW←0
  ・ i←N
  ■ i ＞ j
    ▲ TBL[i] ＜ TBL[i－1]
      ・ WK←TBL[i]
      ・ TBL[i]←TBL[i－1]
      ・ TBL[i－1]←WK
      ・ SW←1
    ▼
    ・ i←i－1
  ■
  ・ j←j＋1
■
```

例　要素数が4のとき
①TBL[1]とTBL[0]を比較
②TBL[2]とTBL[1]を比較
③TBL[3]とTBL[2]を比較
④TBL[2]とTBL[1]を比較
⑤TBL[3]とTBL[2]を比較
⑥TBL[3]とTBL[2]を比較
↑iを使用　↑i－1を使用
バブルソート1では最後の要素から確定するが、バブルソート2では最初の要素から確定する

＜擬似言語演習問題⑬　挿入ソート＞

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。

挿入ソートは整列済み範囲の隣の要素を整列済み範囲の中に挿入して整列させるしょりである
手順は以下の通り
①整列済み範囲(個数)を1とする
②隣(2番目)の要素と1番目の要素を比較･整列
③これで整列済み範囲は2となる
④整列済みの範囲の隣の要素を比較する
⑤整列済みとなるように隣の要素を挿入する
⑥整列済み範囲が広がる
整列済み範囲が配列と同じになるまで繰り返す

［プログラム］

○ プログラム名：　挿入ソート

○ 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←1　**iが整列済み個数を示す**

■ i　＜　N

- j←i－1　**j:挿入対象の要素を示す指標**

■ j　＞　－1

▲ TBL[j]　＜　TBL[j＋1]

- WK←TBL[j]
- TBL[j]←TBL[j＋1]
- TBL[j＋1]←WK

－

- j←0

▼

- j←j－1

■

- i←i＋1

■

下は配列要素｛5,2,3,1,4｝を挿入ソートで整列する例である

	[0]	[1]	[2]	[3]	[4]	
TBL	5	2	3	1	4	5と2を比較
TBL	2	5	3	1	4	交換して整列済み範囲が広がる
TBL	2	5	3	1	4	5と3を比較
TBL	2	3	5	1	4	交換して整列済み範囲が広がる
						(2と5の間に3を挿入したように見える)
TBL	2	3	5	1	4	5と1を比較
TBL	2	3	1	5	4	交換、3と1を比較
TBL	2	1	3	5	4	交換、2と1を比較
TBL	1	2	3	5	4	交換して整列済み範囲が広がる
TBL	1	2	3	5	4	4と5を比較
TBL	1	2	3	4	5	交換して整列済み範囲が広がる

整列済み範囲と配列が同じ大きさになったので終了

（　）が整列済み範囲

＜擬似言語演習問題⑮　スイッチの利用＞

「擬似言語演習問題⑭」のプリントはありません

問　要素数 N の一次元配列 TBL に数値が格納されているとき，要素を先頭から順に SUM に合計する。ただし，－1 が出現した場合は，次の－1 の出現まで合計処理をスキップする。（－1 は必ず偶数個存在するものとする）

TBL	01	02	03	-1	4	5	-1	06	07	08	-1	9	10	11	-1	12
				合計対象外							合計対象外					

上記の例では SUM は，1＋2＋3＋6＋7＋8＋12＝39

［プログラム］

○ プログラム名：　スイッチ

○ 整数型：　N，i，SUM，SW｛N：要素数，SW：スイッチ｝

○ 整数型：　TBL[]｛要素は格納されている｝

・ SUM←0

・ i←0　i:指標

・ SW←0　SWが0のときが合計対象、1のときが合計対象外（初期値は合計対象としておく）

■ i ＜ N

▲ TBL[i] ＝ －1

▲ SW ＝ 1

・ SW←0

・ SW←1　falseでSWを1にしているので、判断は「SW＝1」

▲ SW ＝ 0　合計を計算(↓)するのはSWが0のとき

・ SUM←SUM＋TBL[i]

・ i←i＋1

＜擬似言語演習問題⑯　文字列圧縮＞

問　長さ L の文字列 T（指標：i）について，3 個以上連続する空白を「圧縮した空白を表す文字”#”＋圧縮数」として，圧縮後の文字列を配列 M（指標：j）に格納する。3 個未満の空白はそのままとなる。また，空白が 10 個以上連続していても 1 桁で格納できるものとして，配列 T の最後は空白でないものとする。

例）　連続する空白が 6 個あった場合　→　#6

［プログラム］

○ プログラム名：　文字列圧縮
○ 整数型：　i，j，CNT｛CNT：空白の個数を数える｝
○ 整数型：　L｛L：T の文字列長が格納されている｝
○ 整数型：　T[]，M[]｛配列 T には文字が格納されている｝

・ i←0　配列Tの指標
・ j←0　配列Mの指標
・ M[]←’△’｛配列 M 全体を空白で初期化｝

■ i　＜　L
　・ CNT←0
　■ T[i]　＝　’△’　空白の文字数をカウントする処理
　　・ CNT←CNT＋1
　　・ i←i＋1　順不同
　■
　▲ CNT　≧　3
　　・ M[j]←’#’
　　・ M[j＋1]←CNT
　　・ j←j＋2　#と圧縮数の2文字を格納するので、指標を＋2する
　　・ j←j＋CNT　空白が3文字未満の場合は圧縮せずに文字数分空ける
　▼
　・ M[j]←T[i]　空白以外の文字をコピー
　・ i←i＋1
　・ j←j＋1
■

＜擬似言語演習問題⑰　文字列探索＞

問　長さ L の文字列 S の中に，長さ n の文字列 P が存在するかどうか調べ，存在するならばその S の中での先頭位置を，存在しなければ 0 を POS に格納するものである。

文字列 S（指標：i）　　長さ：L

A	B	X	Y	X	C	Y	X	Y	Z	D	…	F	X	Z	Y
0	1	2	3	4	5	6	7	8	9	10	…	L-4	L-3	L-2	L-1

文字列 P（指標：j）長さ：n

X	Y	Z
0	1	2

この例では POS＝7 となる

［プログラム］

○ プログラム名：　文字列探索
○ 整数型：　i，j，k｛k：文字列 P の先頭文字が一致した位置を保存｝
○ 整数型：　L，n｛L≧n｝
○ 整数型：　S[]，P[]｛要素は格納されている｝
○ 整数型：　POS

「j＜n」の判断について
繰り返しで文字列が見つかった時は「j＝n」となる。
従って繰り返す条件は「j＜n」である。

・ i←0
・ j←0
■ j ＜ n AND L－i ≧ n
この繰り返しが終了するのは、見つかった時か文字列Sを最後まで（この例ではiがL－2になった）検索した時
　・ j←0
　・ k←i
　■ j ＜ n AND S[k] ＝ P[j]
　　・ k←k＋1
　　・ j←j＋1
順不同
　■
　・ i←i＋1
見つかっても違う文字が出ても無条件にiの値を＋1する
■
j ≧ n
j＝n でも可
　・ POS←i－1
見つかった時にiを＋1しているので、－1する
・ POS←0

文字列検索は検索を効率的に行うアルゴリズムが多数ある。その中でもBM法（ボイヤー-ムーア法）が有名なので確認しておこう。

＜擬似言語演習問題⑱　九九表の作成＞

問　二次元配列 TBL に値を設定し，九九表を作成する。ここで，行の指標を i，列の値を j とする。

←列　jを使う

	0	1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7	8	9
1	2	4	6	8	10	12	14	16	18
2	3	6	9	12	15	18	21	24	27
3	4	8	12	16	20	24	28	32	36
4	5	10	15	20	25	30	35	40	45
5	6	12	18	24	30	36	42	48	54
6	7	14	21	28	35	42	49	56	63
7	8	16	24	32	40	48	56	64	72
8	9	18	27	36	45	54	63	72	81

行↑　iを使う

［プログラム］

○ プログラム名：　九九表

○ 整数型：　i，j

○ 整数型：　TBL[，　] ｛TBL は二次元配列，TBL[行，列]とする｝

・ i←0

i ＜ 9

・ i←i＋1　iが0なので先ず＋1する

・ j←0

j ＜ 9

・ j←j＋1　jが0なので先ず＋1する

・ TBL[i-1，j-1]　←i*j

↑配列の指標は0からなので注意

例：4×3の値は3行2列目に格納

行の繰返し

列の繰返し