1　問題

各要素が倍精度実数からなる 4 次元ベクトルに、二乗ノルムによって定まる順序、それが等しい場合は辞書式順序による順序を入れる。具体的に、 2 つのベクトル $x = (x_0, x_1, x_2, x_3)$, $y = (y_0, y_1, y_2, y_3)$ に対して

$$
\begin{aligned}
&x > y \quad iff \\
&\qquad (\sum_{i=0}^{i<4} x_i^2 > \sum_{i=0}^{i<4} y_i^2) or \\
&\qquad (\sum_{i=0}^{i<4} x_i^2 = \sum_{i=0}^{i<4} y_i^2) and \exists j < 4. \{\forall k < j. (x_k = y_k)\} and (x_j > y_j)
\end{aligned}
$$

さらに、以下の記述の便宜上、 $\geq$ を定義する。

$$x \geq y \quad iff \quad x > y or x = y$$

問題は与えられた 4 次元ベクトルのリスト L を昇順にソートすることである。具体的に、 L を並べ替えて L' をつくり、 $i > j \longrightarrow L'[i] \geq L'[j]$ であるようにすること。

ただし、問題として与えるリストの占めるメモリ量はユーザが使用可能な全メモリ量に対して 35% を超えないことを保証する。また、二乗ノルムの計算では、後述の計算方法に従えばオーバーフローを起こさないことを保証する。

2　プログラムの作成

以下、出題者が用意するものと、コンテスト参加者が提出するものを説明する。

2.1　出題者が用意するもの

出題者は以下の関数と、基本的なデータ型を用意する。

C の場合

```
int getlist(void *array, int numelem);
int checkit(int numpe, int *numelem);
int main();
```

`getlist(a, numelem)` は、`(void *)` 型の変数 a の指すアドレスに、高々 `numelem` 個の 4 次元ベクトルからなるリストを返す。さらに、返り値は PE 上に読み込まれるべき要素の個数とする。特に numelem として 0 以下を指定すると、実際にデータが生成されないことに注意せよ。この時に環境変数 `PSC99PROBLEM` を設定することにより、問題の種類を変えることができる。

問題として与えられるリストは、プロセッサの番号の小さい順に上のリストを並べ、一つのリストとみなしたものとする。

Fortran の場合

```
function getlist(array, elem)
double precision array(*)
```

```
integer elem

function checkit(numpe, numelem);
integer numpe, numelem

program main
```

問題として与えられるリストのデータ配置についての注意

4 次元ベクトルは、 4 つの連続した倍精度実数として表現される。 4 次元ベクトルの配列へのアクセス関数は任意に設計してよい。参加者へのヒントとして、いくつかの例を挙げておく。

- 4 次元ベクトルの配列を 2 次元配列と見て、

  ```
   double *list[4];
  ...
  list = malloc( ne * 4 * sizeof(double));
  getlist(list, ne);
  ...
  ```

  とすることができる。これは FORTRAN77 以前でも使える手法である。

  Fortran の場合、例えば以下のようになる。

  ```
  double precision list(4,nmax)
  ...
  getlist(list, ne)
  ...
  ```

- より直接に 1 次元配列とみなすことができる。この場合、リスト`list` の i 番目の 4 次元ベクトルの j 番目の要素には、`list[4*i+j]` でアクセスできる。 FORTRAN の場合も`list(4*i+j)` でアクセスできる。

- 自分でデータ型を定義する方法である。

  ```
  typedef struct {
  double e0;
  double e1;
  double e2;
  dboule e3;
  } vec4;
  ```

  と定義し、

  ```
  vec4 *list;
  ...
  list = malloc( ne * sizeof(vec4));
  getlist(list, ne);
  ...
  ```

とするのは有効なプログラミング手法である。

JIS Fortran においては、以下のようになる。

```
type vec4
 double precision e0
 double precision e1
 double precision e2
 double precision e3
end type vec4
type (vec4) list(nmax)
...
getlist(list, ne)
```

2.2 参加者が提出するもの

参加者側は以下の関数および変数を提出する。

```
void vec4list[];
int numpeatgetlist;
int mypeid();
int psc99(int *numelem);
```

mypeid() は、参加者側で用意するプロセッサの番号付けをする関数であり、 n 個のプロセッサに対して 0 ~ $n-1$ 番までを返す。また、numpeatgetlist にはgetlist() を呼ぶ時点での使用プロセッサ数を定義しておく。これらのセットアップはgetlist を呼び出す時点で参加者側がやっておくこと。

なお、numpeatgetlist は、分散メモリシステムの場合は最大利用可能 PE 数の約数であること。システムが SMP の場合はプロセッサをスレッドと読み替えて理解すること。この場合、上の制約はない。

psc99() は、問題の読み込みから始めて結果を返す関数である。参加者はこれを作成する。関数 psc99(ne) の返り値を n とすると、参加者は n と ne について、以下の条件を満たさなければならない。

$vec4list$ は(void *) 型であるが、(double *) 型と解釈される。そしてプロセッサ $i(0 \leq i < n)$ のアドレス空間にある

```
vec4list[0], vec4list[4], ..., vec4list[4(ne[0]-1)]
```

をプロセッサ番号の小さい順に並べてひとつのリストとみなしたものを答のリストとする。ne[0] に入っている値は、リストを 4 次元ベクトルの配列と解釈した時のリストのサイズである。したがって、倍精度実数の配列と見た時、配列のサイズはne[0] の 4 倍になることに注意せよ。

なお、上記 $vec4list[]$ は mypeid() 関数の返り値のもつプロセッサ j 上のアドレス空間上のポインタと解釈される。なお、これは暗にpsc99 を呼び出した側が、mypeid() 番めのプロセッサ上でvec4list にアクセスができることを要請していることに注意せよ。

上の条件は、例えば main 関数が SPMD スタイルで実行されている時には、自動的に満たされるであろう。なお、共有メモリシステムの場合は、 psc99 は、特定のスレッド (通常 0) でのみ値を返すことが期待されている。その場合はそのスレッド (具体的に main 関数でpsc99 を呼んだスレッド) でのみ条件が満たされていれば良い。

2.3 解答のチェック

出題者は解答が正しく昇順にソートされているかどうかを関数 checkit() を用いてチェックする。checkit() は、mypeid() を呼ぶので、この時点でmypeid() が、有効でなければならない。checkit(n, nelem) は、

実行プロセッサ $i(0 \leq i < n)$ に対して、上のように vec4list に返されたリストが昇順にソートされている場合に 1 そうでない場合に 0 を返す。

なお、チェックする場合に二乗ノルムはベクトルの第一要素の二乗、第二要素の二乗、第三要素の二乗、第四要素の二乗を この順で足して計算する。また、この計算方法を取った場合、オーバーフローを起こさないことを保証する。

2.4 順位の決定

`psc99()` にかかった時間で、順位を決定する。

2.5 その他

今回は問題として与えられるリストのサイズが全システムで同一である保証をしない。システムのメモリサイズによって変化する可能性がある。

使用言語、通信ライブラリその他は一切規定しない。ただし、最終的には出題者側が提供する main 関数は C および Fortran で記述されたものを用意する。

特別の配慮を必要とする言語システムを使った場合は申し出ること。

main の概要は C の場合、以下のようになる。

C の場合

```
int problemid;
extern void *vec4list;

main(int ac, char **av)
{
  int n;
  int ne;

  if (ac == 1) {
    problemid = atoi(getenv("PSC99PROBLEM"));
  } else problemid = atoi(*++av);

  if (problemid < 0 || problemid >= 5) problemid = 0;

/* タイマー 開始 */
  n = psc99(&ne);
/* タイマー 終了 */

if (!checkit(n, &ne))
   fprintf(stdout, "Result is Invalid\n");
}
```

Fortran の場合 ...