1 Problem

Consider vectors of dimension 4, whose component is double precision data. We define the order on them by the euclidean norm, and if the norms are equal, by the lexicographic order. Specifically, let $x = (x_0, x_1, x_2, x_3)$, $y = (y_0, y_1, y_2, y_3)$ be given.

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

We further define $\geq$ for the convenience.

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

The problem is to sort a list of dim-4 vectors in increasing order. Specifically, the problem is to make a permutation L' of L, such that $i > j \longrightarrow L'[i] \geq L'[j]$.

N.B.: We gurantee that the memory occupied by a problem list is equal or less than 35Furthermore, we gurantee that in the calculation of the euclidean norm, an overflow does not occur if calculated by the order designated below.

2 Writing a Program

In the rest of this section, we specify both what we provide, and what you(a contestee) must submit.

2.1 What We Provide

We provide the participants with the following functions.

in the case of C

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

`getlist(a, numelem)` returns a list of at most `numelem` elements to `a` of `(void *)`. Its return value is the number of elements which is to be generated on the processor. Note that specifying numelem less than 0 will result in no generation of data of a problem list. If SMP, "processor" is replaced by "thread." `getlist(a, ne)` returns on a of `(void *)`, the list of the above number of elements. Furthermore, the return value is the same as `getlist(NULL)`.

The problem can be obtained by the environment variable `PSC99PROBLEM`.

The distributed lists are ordered by increasing order on processors to form a problem list.

In the case of Fortran

```
subroutine getlist(array, elem)
double precision array(*)
integer elem

subroutine checkit(result, numpe, numelem);
integer result, numpe, numelem

program main
```

The number of elments of a list generated by `getlist` is returned to `elem`. The result of check by calling `checkit` is returned to `result`.

A Note on the Alignment of Data of a problem list

A vector of dimension 4 is represented as four consecutive double precision numbers. Design of access functions to the list of dim-4 vectors is left to a programmer. Here, we list some typical and useful tips.

- The first tip is to consider arrays of dim-4 vectors as dim-2 arrays as:

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

We list the Fortran version:

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

- More specifically, we can consider them as simply dim-1 arrays. In this case, the j-th element of the i-th vector of the list `list` can be accessed as `list[4*i+j]`. In Fortran, the access method is completely the same.

- The third method is to define a compound(derived) type for dim-4 vectors. Defining vec4 as

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

and using it as

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

are a useful(and modern) programming tip.

In JIS Fortran, we can define a derived type as:

```
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 What You(contestee) must Submit

A contestee must submit the functions and variables below:

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

First a contestee must provide `mypeid()` which returns the processor id. This "processor id" is between 0 and $(n-1)$ for given n processors. In the variable `numpeatgetlist`, a contestee must set the number of processors which call `getlist()`. This initialization must be done by a contestee.

Note that `numpeatgetlist` must be a factor of the maximum number of available processors in the case of distributed memory machines. If SMP, this restriction does not apply.

A contestee must submit the function psc99() which begins in reading a problem list, and sort it. Let the return value of `psc99(ne)` be n. n and ne must satisfy the following condition:

`(void *)` *vec4list* is regarded as `(double *)`, and the solution is the list which is obtained by listing the arrays

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

on the address space of processor $i(0 \leq i < n)$ in increasing order of processor id

The value `ne[0]` is the size of the list when regarding the list as the array of dim-4 vectors. Note that this implies that the size of the array is four times as `ne[0]` when the array is regarded as the 1-dim array of double precisions.

vec4list[] is interpreted as a pointer on the address space on a processor whose id is returned as `mypeid()`. Note that this implicitly requires that the caller of `psc99` can access `vec4list` on `mypeid()`-th processor.

This condition will automatically satisfied when the main function is executed in SPMD style. In the case of SMP, psc99 is expected to return a value on a specific thread. The above condition may be satisfied on the thread.

2.3 Validation of the Answer

A contestee can check wheter the answer is correctly sorted in increasing order by calling `checkit()`. Because `checkit()` calls `mypeid()`, when you call `checkit()`, `mypeid()` must be effective. `checkit(n, nelem)` returns 1 when the list returned in vec4list's of each processor is correctly sorted, or 0 if not.

In the check, `checkit()` calculates the euclidean norm as, first, square the first component, and add the square of the second component to it, then add the square of the third component to it, and then add the square of the fourth component. Moreover, we gurantee that in the calculation of the euclidean norm, an overflow does not occur if calculated by the order designated below.

2.4 Ranking

Ranking, and the Honor is given according to the elapsed time of the submitted function `psc99()`.

2.5 Others

The size of problem lists differ in the memory size of systems.

The contestee can use any programming language, any communication library and any other programming environment.

Note that we provide main function in C and in Fortran. If the contestee uses a programming environemnt which requires special consideration, notify it to us.

The outline of the main function(C) is:

In the case of C

```
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;

/* Timer Start */
  n = psc99(&ne);
/* Timer Stop */

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

In the Case of Fortran ...