1 SUMMARY

Successively arranges a set of real numbers, $\{a_1, a_2, ..., a_n\}$, in order of increasing size using the Heapsort method of J. W. J. Williams. At the k–th stage of the method the k–th smallest member of the set is found. The method is particularly appropriate if it is not known in advance how many smallest members of the set will be required as the Heapsort method is able to calculate the k+1–st smallest member of the set efficiently once it has determined the first k smallest members. The method is guaranteed to sort all n numbers in $O(n\log n)$ operations. If a complete sort is required, the Quicksort algorithm, `KB05,` may be preferred.

ATTRIBUTES — Version: 1.0.0. **Types:** `KB12A, KB12AD.` **Original date:** July 1990. **Origin:** N.I.M. Gould, Rutherford Appleton Laboratory.

2 HOW TO USE THE PACKAGE

2.1 Calling sequence

The method works by rearranging the elements into a so-called “heap removing the smallest element from the heap and then forming another hea with the remaining elements. To form the initial heap, the user must first make a call to `KB12A/AD.` The smallest element in the heap may be recorded and removed, and the remaining elements reordered to form a heap with one fewer element by calling `KB12B/BD.` Further calls t `KB12B/BD` may be used to record further remaining smallest elements.

2.2 Forming the initial heap

Call `KB12A/AD` to form the initial heap.

The single precision version

```
CALL KB12A ( N, A, INDA, LA, INDEX, INFORM )
```

The double precision version

```
CALL KB12AD( N, A, INDA, LA, INDEX, INFORM )
```

`N` is an `INTEGER` variable which must be set by the user to n, the number of entries of `A` which are to be (partially) sorted. It is unchanged by the subroutine. **Restriction:** `N` > 0.

`A` is a `REAL` (`DOUBLE PRECISION` in the D version) array of length `LA` which must be set by the user on input so that its first `N` elements contain the values $a_1, a_2, ..., a_n$. On output, the elements of `A` will have been permuted so that they form a heap.

`INDA` is an `INTEGER` array of length `LA.` If `INDEX` is `.TRUE.` on input, exactly the same permutation is applied to the elements of `INDA` as to the elements of `A.` For example, the permutation will be provided if `INDA(I)` is set to `I,` for `I` = `1, 2, . . . , N,` on entry. If `INDEX` is `.FALSE.` on entry, `INDA` need not be set and will not be altered by the subroutine.

`LA` is an `INTEGER` variable set by the user to actual dimension of the arrays `A` and `INDA.` It is unchanged by the subroutine. **Restriction:** `LA` ≥ `N`.

`INDEX` is a `LOGICAL` variable which must be set `.TRUE.` if the user wishes to sort the index array `INDA` and `.FALSE.` otherwise. It is unchanged by the subroutine.

`INFORM` is an `INTEGER` variable which need not be set by the user on entry to `KB12A/AD.` On exit from `KB12A/AD,` INFORM is used to signal errors in the input data (INFORM > 0) or to indicate a successful call to the routine (INFORM $= 0$). Positive values of INFORM and their meanings are:

1 N was input with a value less than or equal to zero and the heap has not been formed.

2 LA was input with a value less than N and the heap has not been formed.

2.3 Finding the smallest entry in the current heap

Call KB12B/BD to find the smallest entry in a given heap, to place this entry at the end of the list of entries in the heap and th to form a new heap with the remaining entries.

The single precision version

```
CALL KB12B ( M, A, INDA, LA, INDEX, INFORM )
```

The double precision version

```
CALL KB12BD( M, A, INDA, LA, INDEX, INFORM )
```

M is an INTEGER variable which must be set by the user to *m*, the number of entries of A which lie on the heap on entry. It is unchanged by the subroutine. **Restriction:** M > 0.

A is a REAL (DOUBLE PRECISION in the D version) array of length LA which must be set by the user on input so that its first M elements contain the values $a_1, a_2, ..., a_m$, ordered so that the entries lie on a heap. practice, this normally means that they have been placed on a heap by a previous call to KB12A/AD or KB12B/BD. On output, the smallest component of the first M elements of A will have been moved to position A(M) and the remaining elements will now occupy locations $1, 2, ...$ M-1 of A and will again form a heap.

INDA is an INTEGER array of length LA. If INDEX is .TRUE. on input, exactly the same permutation is applied to the elements of INDA as to the elements of A. If INDEX is .FALSE. on entry, INDA need not be set and will not be altered by the subroutine.

LA is an INTEGER variable set by the user to actual dimension of the arrays A and INDA. It is unchanged by the subroutine. **Restriction:** LA $\geq$ M.

INDEX is a LOGICAL variable which must be set .TRUE. if the user wishes to sort the index array INDA and .FALSE. otherwise. It is unchanged by the subroutine.

INFORM is an INTEGER variable which need not be set by the user on entry to KB12B/BD. On exit from KB12B/BD, INFORM is used to signal errors in the input data (INFORM > 0) or to indicate a successful call to the routine (INFORM $= 0$). Positive values of INFORM and their meanings are:

1 M was input with a value less than or equal to zero and the heap has not been reformed.

2 LA was input with a value less than M and the heap has not been reformed.

2.4 Error returns

If some error in the input data is detected by the routines, they return immediately to the calling program with the error flag INFORM having a positive value. See §2.2-2.3 for details.

2.5 Finding the *k* smallest components of a set of *n* elements.

To find the *k* smallest components of a set, $\{a_1, a_2, ..., a_n\}$, of n elements, the user should firstly call KB12A/AD with N $= n$ and a_1 to a_n stored in A(1) to A(N). This places the components of A on a heap. This should then be followed by *k* calls of KB12B/BD, with M $= n - i + 1$ for $i = 1, ..., k$. The required *k* smallest values, in increasing order of magnitude, will now occupy positions $n - i + 1$ of A for $i = 1, ..., k$.

3 GENERAL INFORMATION

Use of common: None.

Workspace: None.

Other routines called directly: None.

Input/output: None.

Restrictions: `LA > N > 0 (KB12A/AD)` and `LA > M > 0 (KB12B/BD)`.

4 METHOD

The Heapsort method is due to J. W. J. Williams (Algorithm 232, Communications of the ACM **7** (1964), 347-348). Subroutine `KB12A/AD` is a partial amalgamation of Williams' Algol procedures *setheap* and *inheap* while `KB12B/BD` is based upon his procedures *outheap* and *swopheap.*

The elements of the set $\{a_1, a_2, ..., a_n\}$ are first allocated to the nodes of a heap. A heap is a binary tree in which the element at each parent node has a smaller numerical value than the elements at its two children. The smallest value is thus placed at the root of the tree. This value is now removed from the heap and a subset of the remaining elements interchanged until a new, smaller, heap is constructed. The smallest value of the new heap is now at the root and may be removed as described above. The elements of the initial set may thus be arranged in order of increasing size, the *i*–th largest element of the array being found in the *i*–th sweep of the method. The method is guaranteed to sort all *n* numbers in $O(n\log n)$ operations.

5 EXAMPLE OF USE

As a simple example, suppose we wish to find the 12 smallest elements of the set $\{a_1, a_2, ..., a_{20}\}$ = $\{-5, -7, 2, 9, 0, -3, 3, 5, -2, -6, 8, 7, -1, -8, 10, -4, 6, -9, 1, 4\}$.

```
      PROGRAM MAIN
      INTEGER N, LA, I, M, INFORM
      PARAMETER ( N = 20, LA = N )
      LOGICAL INDEX
      INTEGER INDA( LA )
      DOUBLE PRECISION A( LA )
      EXTERNAL KB12AD, KB12BD
      DATA A / -5.0, -7.0, 2.0, 9.0, 0.0, -3.0, 3.0, 5.0, -2.0, -6.0,
     *         8.0, 7.0, -1.0, -8.0, 10.0, -4.0, 6.0, -9.0, 1.0, 4.0 /
      DATA INDA / 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,
     *            15, 16, 17, 18, 19, 20 /
      INDEX = .TRUE.
C
C  BUILD THE HEAP.
C
      CALL KB12AD( N, A, INDA, LA, INDEX, INFORM )
C
C  REORDER THE VARIABLES.
C
      DO 10 I = 1, 12
         M    = N - I + 1
         CALL KB12BD( M, A, INDA, LA, INDEX, INFORM )
         WRITE( 6, 2000 ) I, INDA( M ), A( M )
   10 CONTINUE
      STOP
 2000 FORMAT( ' The ', I2, '-th(-st) smallest value, a(', I2,
     *        '), is ', 1P, D12.4 )
      END
```

This produces the following output:

```
 The  1-th(-st) smallest value, a(18), is  -9.0000D+00
 The  2-th(-st) smallest value, a(14), is  -8.0000D+00
 The  3-th(-st) smallest value, a( 2), is  -7.0000D+00
 The  4-th(-st) smallest value, a(10), is  -6.0000D+00
 The  5-th(-st) smallest value, a( 1), is  -5.0000D+00
 The  6-th(-st) smallest value, a(16), is  -4.0000D+00
 The  7-th(-st) smallest value, a( 6), is  -3.0000D+00
 The  8-th(-st) smallest value, a( 9), is  -2.0000D+00
 The  9-th(-st) smallest value, a(13), is  -1.0000D+00
 The 10-th(-st) smallest value, a( 5), is   0.0000D+00
 The 11-th(-st) smallest value, a(19), is   1.0000D+00
 The 12-th(-st) smallest value, a( 3), is   2.0000D+00
```