A radix sort sorts a list of identifiers. An iterative process is used to sort the list. The number of iterations is equal to the number of characters in the longest identifier. Sorting begins with the rightmost or least significant character in the identifier. A bucket is created for each character in the alphabet of characters used to make identifiers. The buckets are placed in lexicographic order. Identifiers are placed in the bucket associated with their rightmost character in the first iteration. After identifiers have been placed in buckets, a new list is created. The new list is partially ordered. The new list consists of the identifiers in the buckets concatenated in lexicographic order. For the second iteration, the next to last character in the identifier is selected. The foregoing process is repeated, assigning identifiers to buckets according to their penultimate character and reassembling the list by concatenating the buckets in lexicographic order.

Consider the following examples:

Iteration 1, rightmost character.
List: 64, 8, 216, 512, 27, 729, 0, 1, 343, 125

Bucket	Identifier
0	0
1	1
2	512
3	343
4	64
5	125
6	216
7	27
8	8
9	729

Iteration 2, second character from rightmost.
List: 0, 1, 512, 343, 64, 125, 216, 27, 8, 729

Bucket	Identifier
0	0, 1, 8
1	216, 512
2	729, 27, 125
3	
4	343
5	
6	64
7	
8	
9	

Iteration 3, leftmost character.
List: 0, 1, 8, 216, 512, 729, 27, 125, 343, 64

Bucket	Identifier
0	0, 1, 8, 27, 64
1	125
2	216
3	343
4	
5	512
6	
7	729
8	
9	

Sorted List: 0, 1, 8, 27, 64, 125, 216, 343, 512, 729

Note that the rightmost character is used in the first iteration to place identifiers in buckets. For example, identifier 64 is placed in bucket 4. Identifiers having fewer than three characters are assumed to have leading zeros. For example, the identifier 1, is assumed to be 001.

The list is reconstituted for iteration 2. The identifiers are reordered so they appear in the order in which the buckets are ordered. Bucket 0 is first so the identifier in bucket zero is first on the list. Subsequent identifiers are placed on the list in the same order as they occur in the buckets.

Note that the middle character is used to sort identifiers in the second iteration. Identifier 512 is placed in the bucket marked 1.

The list is reconstructed for iteration 3 as it was for the second iteration.

Another example:
List: mat, an, ant, a, am, tan, ten, en, em, met

Rule: if length(id)<position put id in bucket α.

Position=3	
List: mat, an, ant, a, am, tan, ten, en, em, met	
Bucket	**Identifier**
a	an, a, am, en, em
a	
e	
m	
n	tan, ten
t	mat, ant, met

Position=2	
List: an, a, am, en, em, tan, ten, mat, ant, met	
Bucket	**Identifier**
a	a
a	tan, mat
e	ten, met
m	am, em
n	an, en, ant
t	

Position=1	
List: a, tan, mat, ten, met, am, em, an, en, ant	
Bucket	**Identifier**
a	
a	a, am, an, ant
e	em, en
m	mat, met
n	
t	tan, ten

Sorted list: a, am, an, ant, em, en, mat, met, tan, ten

Radix sort
void *Radix*::*SortMgr*(*istream*& *i*, *ostream*& *o*)
1. Declare list *L*
2. Read the identifiers in stream *i* into list *L*. Use member function *TaiIinsert* to put the identifiers in the list.
3. Declare integer *p*. Variable *p* is the character position that is used to select the bucket where an identifier is inserted
4. Declare variable *length* and initialize it to one less than the length of the longest identifier in the list.
5. **for** *p=length* **downto** 0 **do** *BucketSort*(*l*,*p*);
6. Write list *L* output stream *o*.

void *BucketSort*(*List*& *L*, **int** *p*)
1. **while** list *L* is not empty **do**
   1.1. Use member function *HeadRemove* to remove element *e* from the head of list *L*.
   1.2. **if** the length of the identifier in element e is shorter than position p **then**
      1.2.1. use member function *TailInsert* to append element *e* on the list for the alpha bucket
   1.3. **else**
      1.3.1. use member function *TailInsert* to append element *e* on the list for the bucket whose index is given by the integer code the corresponds the p^{th} letter of the identifier in element *e*.
2. Use member function *Join* to join all the buckets to list *L*.