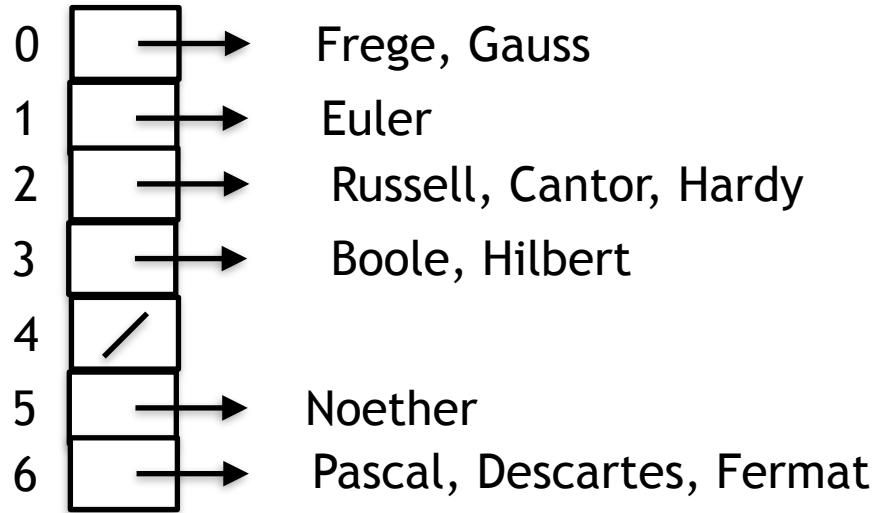


# Homework 1

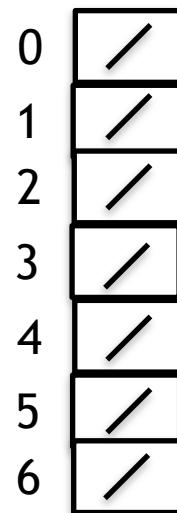
- Symbol Table
- Implement the functions defined in SymTab.h in a file called SymTab.c
- Do not change the contents of SymTab.h
- Implement a driver program to test your implementation
- “Due” Monday September 21

# Separate Chaining



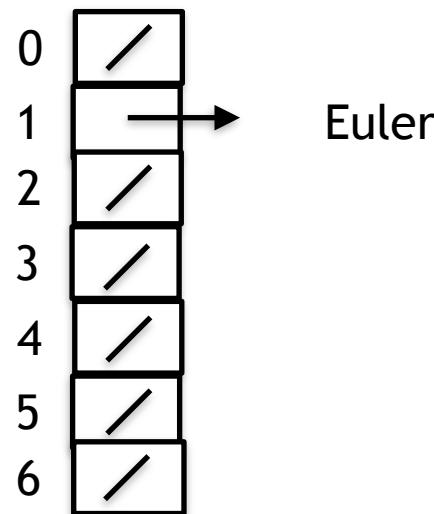
# Separate Chaining Example

Key	Hash Function Value
Euler	1
Frege	0
Russell	2
Pascal	6
Boole.	3
Descartes	6
Gauss	0
Noether	5
Cantor	2
Fermat	6
Hardy	2
Hilbert	3



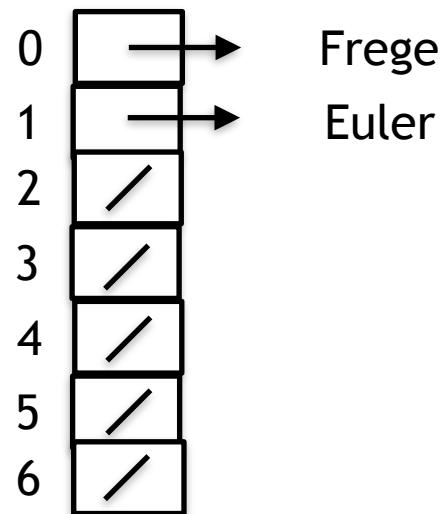
# Separate Chaining Example

Key	Hash Function Value
Euler	1
Frege	0
Russell	2
Pascal	6
Boole.	3
Descartes	6
Gauss	0
Noether	5
Cantor	2
Fermat	6
Hardy	2
Hilbert	3



# Separate Chaining Example

Key	Hash Function Value
Euler	1
Frege	0
Russell	2
Pascal	6
Boole.	3
Descartes	6
Gauss	0
Noether	5
Cantor	2
Fermat	6
Hardy	2
Hilbert	3



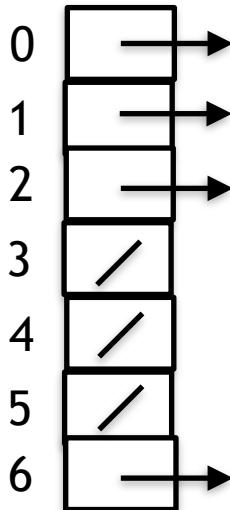
# Separate Chaining Example

Key	Hash Function Value		
Euler	1		
Frege	0		
Russell	2		
Pascal	6		
Boole.	3		
Descartes	6		
Gauss	0		
Noether	5		
Cantor	2		
Fermat	6		
Hardy	2		
Hilbert	3		

The diagram illustrates a separate chaining hash table with 7 slots. Slots 0, 1, and 2 contain pointers to external boxes labeled Frege, Euler, and Russell respectively. Slots 3, 4, 5, and 6 each contain a diagonal slash.

# Separate Chaining Example

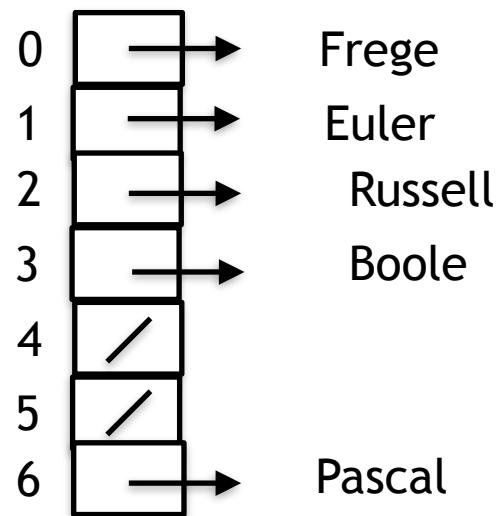
Key	Hash Function Value	
Euler	1	
Frege	0	
Russell	2	
Pascal	6	
Boole.	3	
Descartes	6	
Gauss	0	
Noether	5	
Cantor	2	
Fermat	6	
Hardy	2	
Hilbert	3	



The diagram illustrates a separate chaining hash table with 7 slots. Slots 0, 1, 2, 4, and 6 each point to an external box containing a name: Frege, Euler, Russell, Descartes, and Pascal respectively. Slots 3, 5, and 6 contain internal diagonal slashes, indicating they are part of a linked list.

# Separate Chaining Example

Key	Hash Function Value
Euler	1
Frege	0
Russell	2
Pascal	6
Boole.	3
Descartes	6
Gauss	0
Noether	5
Cantor	2
Fermat	6
Hardy	2
Hilbert	3



# Separate Chaining Example

Key	Hash Function Value	
Euler	1	
Frege	0	
Russell	2	
Pascal	6	
Boole.	3	
Descartes	6	
Gauss	0	
Noether	5	
Cantor	2	
Fermat	6	
Hardy	2	
Hilbert	3	

The diagram illustrates a separate chaining hash table with 7 slots. Each slot is represented by a vertical rectangle with a horizontal pointer at its right end. The slots are indexed from 0 to 6 on the left. The pointers indicate the head of a linked list for each slot:

- Slot 0 points to Frege
- Slot 1 points to Euler
- Slot 2 points to Russell
- Slot 3 points to Boole
- Slot 4 contains a diagonal slash (/) and points to Pascal
- Slot 5 contains a diagonal slash (/) and points to Descartes
- Slot 6 points to Pascal and Descartes (both Pascal and Descartes are listed here)

# Separate Chaining Example

Key	Hash Function Value		
Euler	1		
Frege	0		
Russell	2		
Pascal	6		
Boole.	3		
Descartes	6		
Gauss	0		
Noether	5		
Cantor	2		
Fermat	6		
Hardy	2		
Hilbert	3		

The diagram illustrates a separate chaining hash table with 7 slots. Each slot is represented by a vertical rectangle with a horizontal pointer at its right end. The slots are indexed from 0 to 6. The pointers indicate the head of a linked list for each slot:

- Slot 0: Points to Frege, Gauss
- Slot 1: Points to Euler
- Slot 2: Points to Russell
- Slot 3: Points to Boole
- Slot 4: Contains a diagonal slash (/) and points to Pascal, Descartes
- Slot 5: Contains a diagonal slash (/) and points to Pascal, Descartes
- Slot 6: Points to Pascal, Descartes

# Separate Chaining Example

Key	Hash Function Value		
Euler	1		
Frege	0		
Russell	2		
Pascal	6		
Boole.	3		
Descartes	6		
Gauss	0		
Noether	5		
Cantor	2		
Fermat	6		
Hardy	2		
Hilbert	3		

The diagram illustrates a separate chaining hash table with 7 slots. Each slot is represented by a small square box containing a number from 0 to 6. A horizontal line extends from the right side of each box, ending in an arrow that points to the name of a mathematician. The mapping is as follows: Slot 0 points to 'Frege, Gauss'; Slot 1 points to 'Euler'; Slot 2 points to 'Russell'; Slot 3 points to 'Boole'; Slot 4 contains a diagonal slash and has no pointer; Slot 5 points to 'Noether'; and Slot 6 points to 'Pascal, Descartes'.

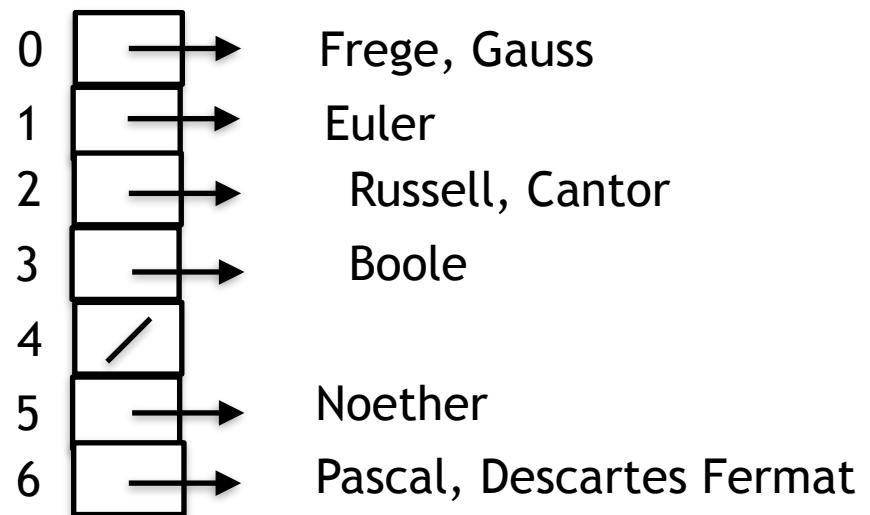
# Separate Chaining Example

Key	Hash Function Value		
Euler	1		
Frege	0		
Russell	2		
Pascal	6		
Boole.	3		
Descartes	6		
Gauss	0		
Noether	5		
Cantor	2		
Fermat	6		
Hardy	2		
Hilbert	3		

The diagram illustrates a separate chaining hash table with 7 slots. Each slot is represented by a small square box containing a number from 0 to 6. A horizontal line extends from the right side of each box, ending in an arrow that points to the name of a mathematician. The mapping is as follows: Slot 0 points to 'Frege, Gauss'; Slot 1 points to 'Euler'; Slot 2 points to 'Russell, Cantor'; Slot 3 points to 'Boole'; Slot 4 contains a diagonal slash and has no pointer; Slot 5 points to 'Noether'; and Slot 6 points to 'Pascal, Descartes'.

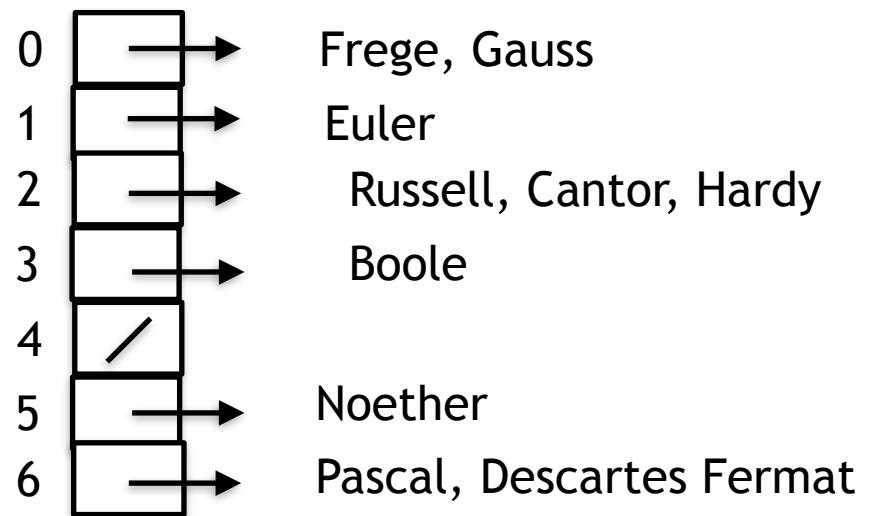
# Separate Chaining Example

Key	Hash Function Value
Euler	1
Frege	0
Russell	2
Pascal	6
Boole.	3
Descartes	6
Gauss	0
Noether	5
Cantor	2
Fermat	6
Hardy	2
Hilbert	3



# Separate Chaining Example

Key	Hash Function Value
Euler	1
Frege	0
Russell	2
Pascal	6
Boole.	3
Descartes	6
Gauss	0
Noether	5
Cantor	2
Fermat	6
Hardy	2
Hilbert	3



# Separate Chaining Example

Key	Hash Function Value		
Euler	1		
Frege	0		
Russell	2		
Pascal	6		
Boole.	3		
Descartes	6		
Gauss	0		
Noether	5		
Cantor	2		
Fermat	6		
Hardy	2		
Hilbert	3		

Diagram illustrating Separate Chaining:

The diagram shows a vertical array of 7 slots, each represented by a rectangle with an arrow pointing to its corresponding value from the table above. Slot 4 is empty and contains a diagonal slash.

- Slot 0: Frege, Gauss
- Slot 1: Euler
- Slot 2: Russell, Cantor, Hardy
- Slot 3: Boole, Hilbert
- Slot 4: / (empty)
- Slot 5: Noether
- Slot 6: Pascal, Descartes, Fermat

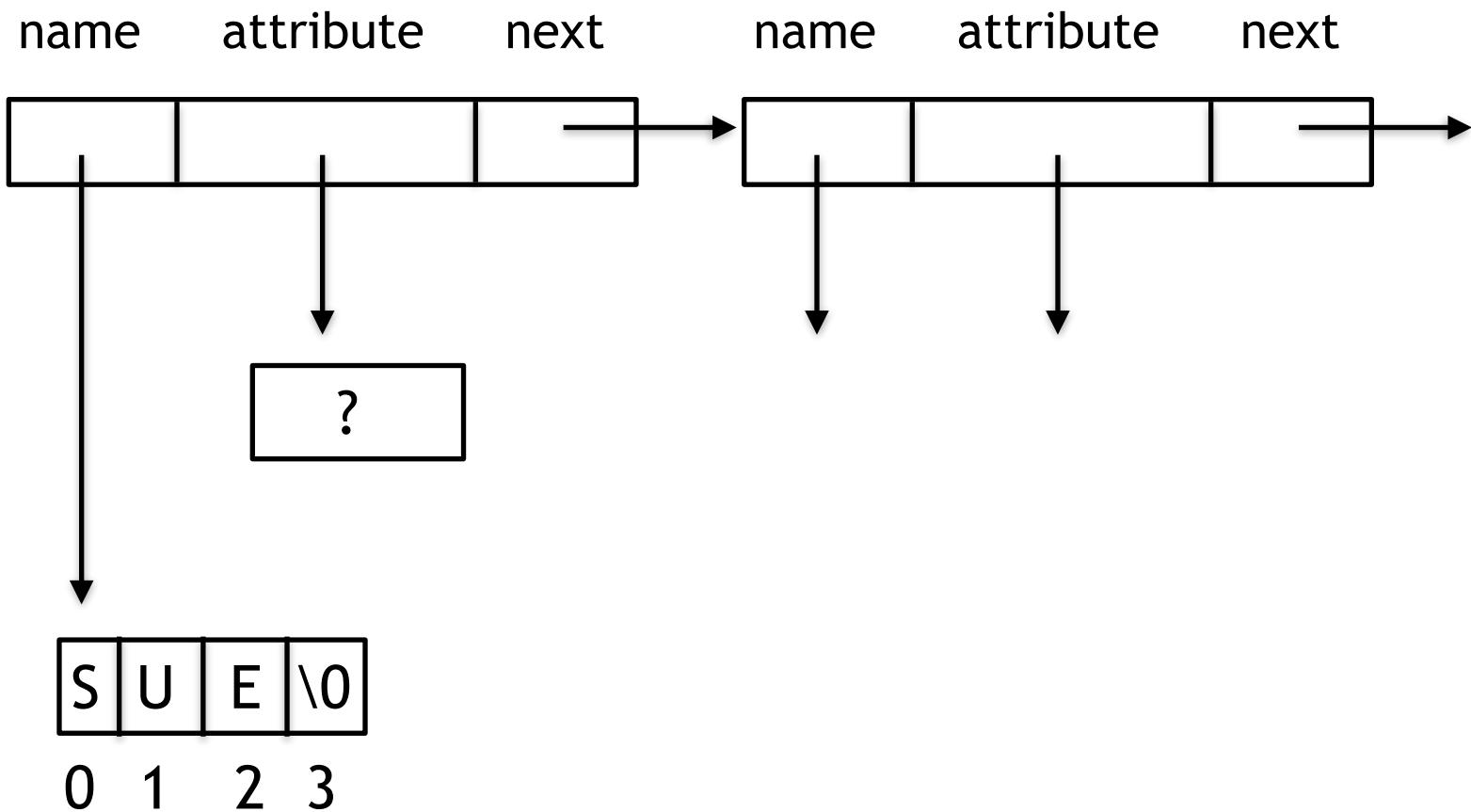
# SymTab.h

```
/*
API for a symbol table. The symbol table stores (name, attribute) pairs. The data
type for the attribute is void * so programs that use the symbol table can
associate any attribute type with a name

The symbol table is implemented using a separate chaining hash table.

*/
//A SymEntry is the building block for linked lists of (name, attribute) pairs
typedef struct SymEntry {
    char * name;
    void * attribute;
    struct SymEntry * next;
} SymEntry;
```

# SymEntry



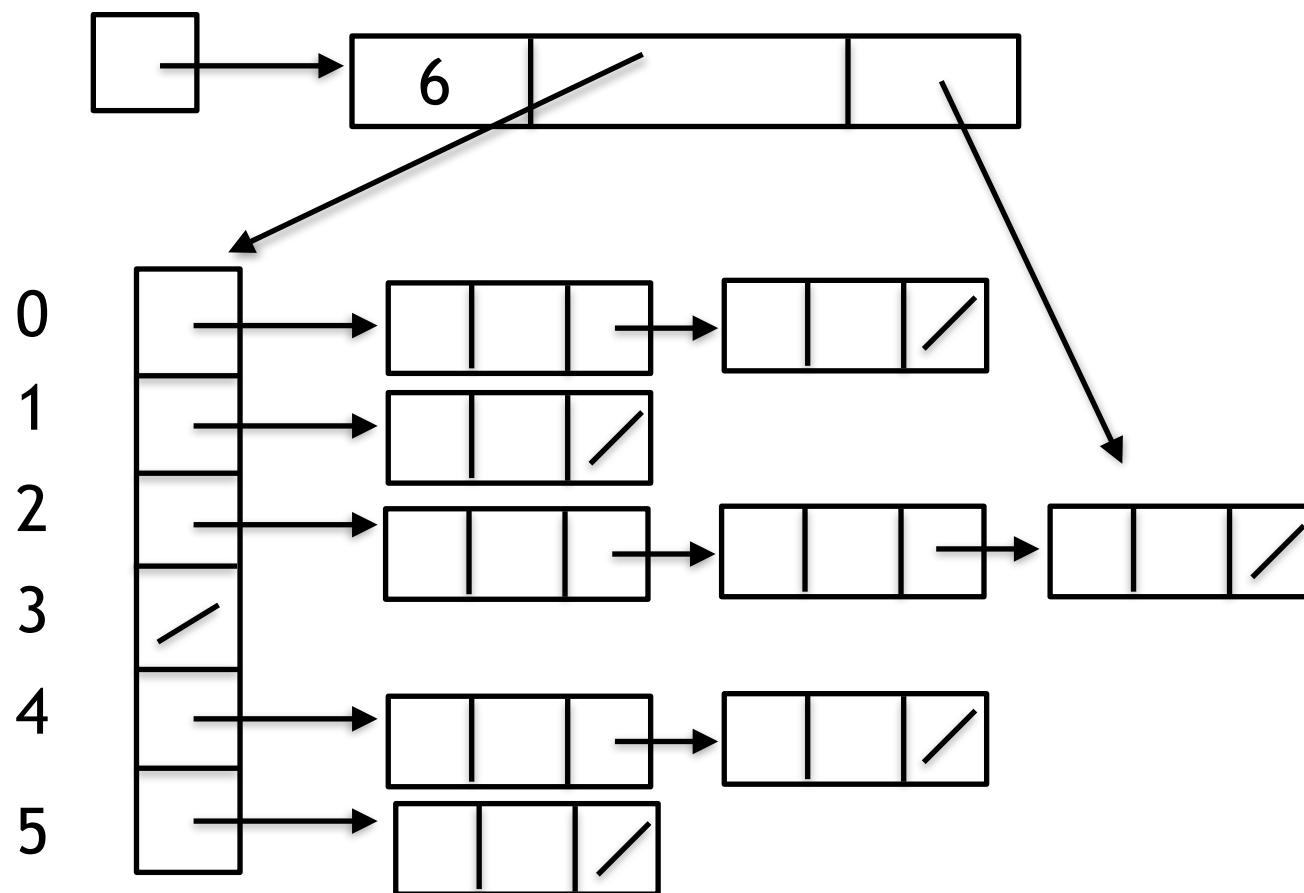
# SymTab.h

```
/*
  Each symbol table is represented by a SymTab
  size is the current number of lists in the separate chaining hash table
  contents is an array of lists (i.e. points to the zeroth element in the array)
  if current is not NULL it points to the current (name,attribute)
    pair in the symbol table
*/
typedef struct {
    int size;
    SymEntry **contents;
    SymEntry *current;
} SymTab;

SymTab * createSymTab(int size);
/* PRE: size >= 0
   size is an estimate of the number of items that will be stored in the symbol
   table
   Return a pointer to a new symbol table
*/
```

# SymTab

SymTab \* size contents current



# SymTab.h

```
//In the following functions assume a pre condition that table references a
//previously created symbol table

void destroySymTab(SymTab *table);
//recover space created by the symbol table functions
//no functions should use the symbol table after it is destroyed

int enterName(SymTab * table, char *name);
/*if name is not in the symbol table, a copy of name is added to the symbol table
   with a NULL attribute, set current to reference the new (name, attribute) pair
   and return 1

   if name is in the symbol table, set current to reference the (name, attribute)
   pair and return 0
*/
int findName(SymTab *table, char *name);
/*if name is in the symbol table, set current to reference the (name, attribute)
   pair and return 1
   otherwise do not change current and return 0
*/
```

# SymTab.h

```
int hasCurrent(SymTab *table);
//if current references a (name, attribute) pair return 1
//otherwise return 0;

void setCurrentAttr(SymTab *table, void * attr);
//PRE: hashCurrent() == 1
//change the attribute value of the current (name, attribute) pair to attr

void * getCurrentAttr(SymTab *table);
//PRE: hasCurrent() == 1
//return the attribute in the current (name, attribute) pair

char * getCurrentName(SymTab *table);
//PRE: hasCurrent() == 1
//return the name in the current (name, attribute) pair
```

# SymTab.h

```
//Assume no changes are made to the symbol table while iterating through the symbol table

int startIterator(SymTab *table);
//if the symbol table is empty, return 0
//otherwise set current to the "first" (name, attribute) pair in the symbol table and return 1

int nextEntry(SymTab *table);
/*if all (name, attribute) pairs have been visited since the last call to
 startIterator, return 0
 otherwise set current to the "next" (name, attribute) pair and return 1
*/
```