CEng 242 Homework 4

Due: 28^{th} April 2001

In this homework you will implement a C++ class called `AssocArray` which is a mapping from `char *` type to `double`. An object of `AssocArray` will keep a maximum number of elements indexed by `char *` values. Following member functions will be implemented in the `public` part:

`AssocArray(int m)`
: This constructor function will set maximum number of elements in the associative array to be '`m`'. There is no upper limit for value of '`m`'. That means you should allocate array storage dynamically. If given m is less than or equal to 0, execute the command `throw "Invalid size";`.

`~AssocArray()`
: Destructor function should deallocate all dynamic memory allocated by the object.

`int exists(const char* s)`
: Searches if the element with index '`s`' in the object. Returns true (integer 1) if element is found, otherwise false (integer 0).

`double & operator[](const char *s)`
: Searches for the element with index '`s`' in the object. If it exists returns a reference to double content of it. Otherwise inserts a new element initialized to `0.0` and returns a reference to it. If the array is full — current number of elements is equal to the size of the array in the construction — execute the command `throw  "array overflow";`.

`void erase(const char *s)`
: Searches the element with index '`s`' and deletes it. If element does not exist, execute the command `throw "element is not found";`.

`AssocArray & operator*(AssocArray &right)`
: This operator takes the intersection of two arrays where only element indexes existing in both objects (current and `right`) will be kept in the resulting object. `double` value of the element in the resulting object will be set to the value comming from `right` object. Note that you should create a new object and return the reference of it. Size of the new object should be set to minimum of two objects. Do not modify values of current object or `right` object.

`AssocArray & operator+(AssocArray &right)`
: This operator takes the union of two arrays where unique elements in both objects are directly inserted in the resulting object and elements existing in both objects are inserted once. Resulting object should contain the value from the `right` object if element exists in both objects. You should create a new object and size of the object should be sum of the sizes of two objects. Do not modify current or `right` object.

`AssocArray & operator-(AssocArray &right)`
: This opreator takes the set difference of two objects. Resulting object will contain all elements from current object except the elements already existing in `right` object. You should create a new object and size of the object should be same with the size of the current object.

`AssocArray & operator=(AssocArray &right)`
: This operator is the assignment operator where the data in the `right` operator is copied element by element to current object and a reference to current object is returned. Size of the current object is set to the size of the `right` object, dynamic storage is adjusted and no garbage should be produced.

`void iterate(void (*f)(const char*,AssocArray &))`

This function is a generic iterator for elements of the array. It calls the function parameter '`f`' for all element indexes where element index is the first parameter for `f` and object itself is the second parameter. '`f`' should be called by increasing order of element index strings.

For example the following function:

```
void echo(const char* s,AssocArray &O) {
   cout << s << ':' << O[s] << '\n';
}
```

will print all element-value pairs in the object `A` if '`A.iterate(echo);`' is called.

Hide all implementation details including auxilary functions in private section of the object and do not declare any other thing (types, functions) than the `AssocArray`.
Index strings can be at most 25 characters long. If a larger string is given ignore the characters after the 25^{th}. For example '`"abcdefghijklmnopqrstuvwxyz"`','`"abcdefghijklmnopqrstuvwxyabcd"`' and '`"abcdefghijklmnopqrstuvwxy"`' will be considered to be equal (to the last one). All string comparisons are case sensitive.
In your submission only provide a header file `assocarray.h` containing your `AssocArray` declaration and necessary includes from C,C++ libraries.

Example:

```
#include<iostream.h>
#include"assocarray.h"
void echo(const char* s,AssocArray &O) {
   cout << s << ':' << O[s] << '\n';
}
void clear(const char *s,AssocArray &O) {
   O.erase(s);
}
int main() {
    AssocArray A(10),B(5),C(10);

    A["ahmet"]=3.33;
    A["mehmet"]=3.00;
    A["ali"]=2.00;
    A["ayse"]=2.80;
    B["ahmet"]=2.33;
    B["ayse"]=3.00;
    A["ali"] += 0.5;
    cout << A["onur"] <<'\n';
    C=A+B; C.iterate(echo);
    C=A*B; C.iterate(echo);
    C=A-B; C.iterate(echo);
    if (A.exists("onur"))
      A.erase("onur");
    if (A.exists("ahmet"))
      A.erase("ahmet");
    A.iterate(echo);
    C.iterate(clear);
    return 0;
}
```