CS232 Lab4
sp2023
streller

PROBLEM BACKGROUND:

A table is an unordered group of “records” or key-value pairs each of which has a designated field called the key field. Records within the table are identified by the value of the key field. The table is an appropriate ADT for information retrieval systems in which individual records must be accessed frequently (and quickly) but the entire collection of records must never be processed sequentially. Synonymous terms are lookup table and map. The latter stresses that a table is really a mapping.
A simple example of a table is the dictionary: the word is the key and the definition is the value.
Read TableHandout.pdf under the CS232 Notes section of the course webpage for more.

PROBLEM STATEMENT:

Design a table ADT
Specifically write a table class.
It must be generic (i.e. templated) and contain all the necessary attributes and actions (data members and member functions) for the table class to behave as expected.

You must write a table.t file that works for the provided files table.h pair.h pair.cpp driver.cpp.
(you will have to create tableSpecificOps1.h, and .cpp and table.cpp as well)

See the window captures below for a running program

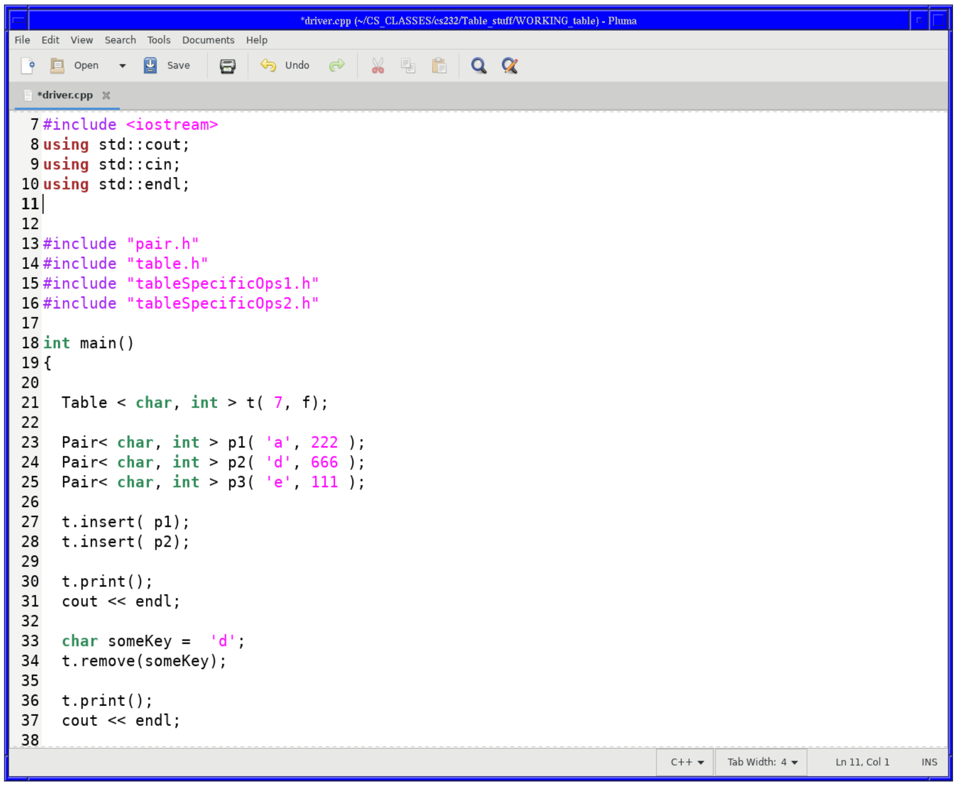
DELIVERABLES:

hard:	table.t all stapled of course
soft	in a zipped file, called CS232_Lab4_yourLastName, submitted to brightspace cs232_Lab4Ans table. t and tableSpecificOps1.h
due	8:00am, 14 February 2023

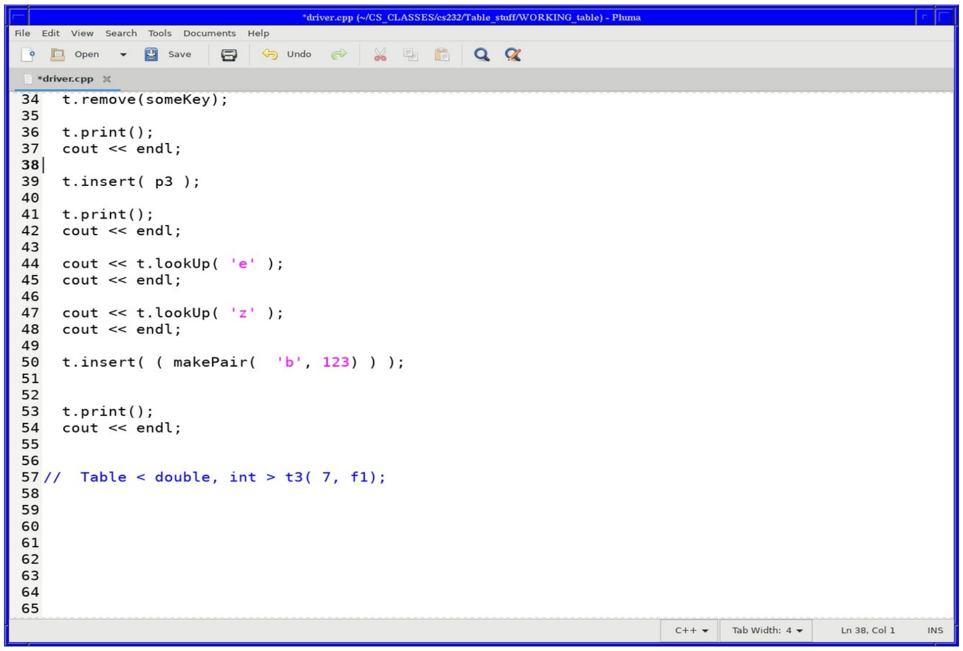
grading will be based on:

1. table class design consisting of all necessary components
2. table class compiling
3. the program running successfully with all the above mentioned files

Example working driver for the table class

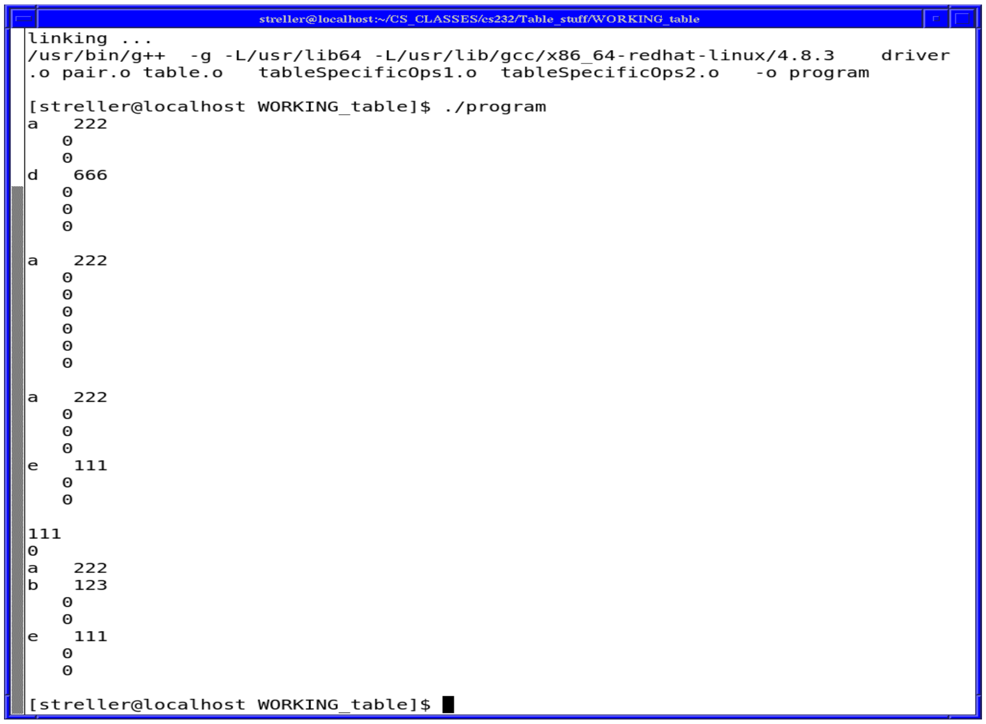

*driver.cpp (~/CS_CLASSES/cs232/Table_stuff/WORKING_table) - Pluma
File Edit View Search Tools Documents Help
Open
Save
Undo
*driver.cpp
C++
Tab Width: 4
Ln 11, Col 1
INS

```cpp
#include <iostream>
using std::cout;
using std::cin;
using std::endl;


#include "pair.h"
#include "table.h"
#include "tableSpecificOps1.h"
#include "tableSpecificOps2.h"

int main()
{

  Table < char, int > t( 7, f);

  Pair< char, int > p1( 'a', 222 );
  Pair< char, int > p2( 'd', 666 );
  Pair< char, int > p3( 'e', 111 );

  t.insert( p1);
  t.insert( p2);

  t.print();
  cout << endl;

  char someKey =  'd';
  t.remove(someKey);

  t.print();
  cout << endl;

```


*driver.cpp (~/CS_CLASSES/cs232/Table_stuff/WORKING_table) - Pluma
File Edit View Search Tools Documents Help
Open
Save
Undo
*driver.cpp
C++
Tab Width: 4
Ln 38, Col 1
INS

```cpp
  t.remove(someKey);

  t.print();
  cout << endl;

  t.insert( p3 );

  t.print();
  cout << endl;

  cout << t.lookUp( 'e' );
  cout << endl;

  cout << t.lookUp( 'z' );
  cout << endl;

  t.insert( ( makePair(  'b', 123) ) );


  t.print();
  cout << endl;


//  Table < double, int > t3( 7, f1);








```

output from driver



streller@localhost:~/CS_CLASSES/cs232/Table_stuff/WORKING_table

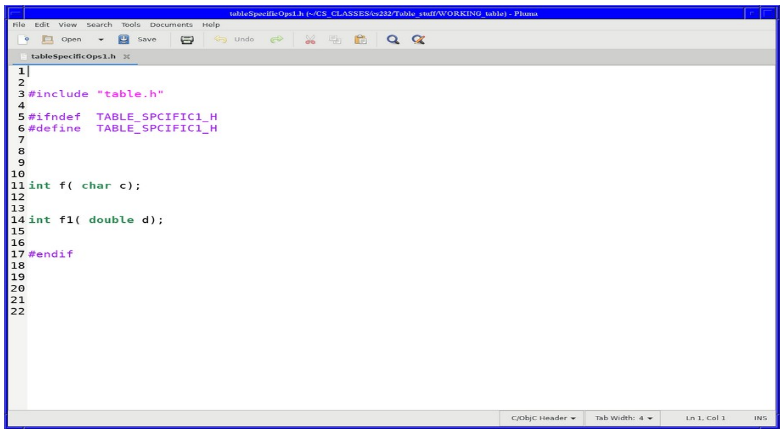
```
linking ...
/usr/bin/g++  -g -L/usr/lib64 -L/usr/lib/gcc/x86_64-redhat-linux/4.8.3    driver
.o pair.o table.o   tableSpecificOps1.o  tableSpecificOps2.o   -o program

[streller@localhost WORKING_table]$ ./program
a   222
   0
   0
d   666
   0
   0
   0

a   222
   0
   0
   0
   0
   0
   0

a   222
   0
   0
   0
e   111
   0
   0

111
0
a   222
b   123
   0
   0
e   111
   0
   0

[streller@localhost WORKING_table]$ 
```

the table constructor populates the table with default values of 0.
the printing is only for development/debugging purposes and would be removed from final release

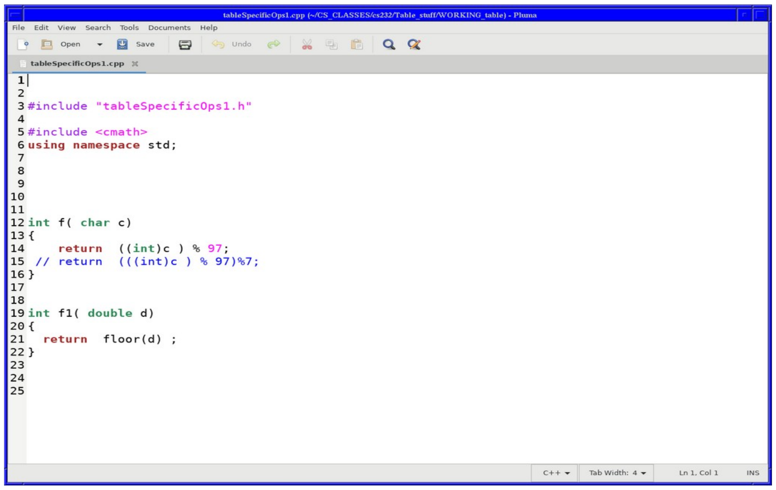
Note: when declaring the table: f is the mapping passed in. f is a variable, which changes from application to application

```
Table < char, int > t( 7, f);
```

In this example, the mapping f is defined in a separate file called "tableSpecificOps1.h"


tableSpecificOps1.h (~/CS_CLASSES/cs232/Table_stuff/WORKING_table) - Pluma
File Edit View Search Tools Documents Help
Open Save Undo
tableSpecificOps1.h
C/ObjC Header Tab Width: 4 Ln 1, Col 1 INS

```


#include "table.h"

#ifndef  TABLE_SPCIFIC1_H
#define  TABLE_SPCIFIC1_H




int f( char c);


int f1( double d);


#endif





```


tableSpecificOps1.cpp (~/CS_CLASSES/cs232/Table_stuff/WORKING_table) - Pluma
File Edit View Search Tools Documents Help
Open Save Undo
tableSpecificOps1.cpp
C++ Tab Width: 4 Ln 1, Col 1 INS

```


#include "tableSpecificOps1.h"

#include <cmath>
using namespace std;





int f( char c)
{
    return  ((int)c ) % 97;
 // return  (((int)c ) % 97)%7;
}


int f1( double d)
{
  return  floor(d) ;
}



```

Of course the developer must determine what the actual mapping is; in this example that's f

NOTE:

All submitted code for this course must use the separation model for code layout.
(ie .h, .cpp and where applicable .t files)
See interfaces.pdf under the cs232 notes section of the course webpage

Code not in this format will receive an automatic 20 point deduction

Also all code must have the appropriate file comment header and function comment header as spelled out in the documentation standards handbook.

Acceptable alternatives are

```
/*****************************************************************************************
*
*   File name:     xxx.extention
*
*   This file contains ....
*    Or The purpose is blah, blah, etc ...
*
*   Programmer:
*
*   Date Written:
*
*   Date Last Revised:
*
*****************************************************************************************/
```

And

```
/*****************************************************************************************
*
*   Function Name:
*
*   Purpose:
*
*
*   Input Parameters:
*
*   Output parameters:
*
*   Return Value:
*
*****************************************************************************************/
```

Note the order of presentation.
This MUST be followed.

Code not in this format will receive an automatic 20 point deduction