1. Download the following zipped file. It contains project and data files.

<https://markbowman.org/231/Lab28.zip>

2. You will make changes to Lab28.cpp, but not Node.h or Node.cpp

3. Create a workspace and compile and run the program. You should see the output below:

Enter file name: ***Lab28a.txt***

0

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

Collisions: 0

4. Modify the input loop in your main() function to do the following:

* Assign the pointer p to a new node for each line of input.
* Call the hash\_insert() function to set one of the entries in table[] to p.
* Use the value returned from hash\_insert() to update count.

After the input loop completes, the node table[] should have values pointing to each node created.

5. Create the hash\_insert() function, using the declaration provided. Your function will use the hash function:

*index = (ID \* 215 + 37) mod 21*

It should calculate the index and resolve collisions:

* If an entry is in use, loop to increment the index until an open index is found.
* If your loop gets to the end of the table, go back to the first entry.

The function should return the number of collisions for each insert attempt.

Hint - test to see if a table entry is being used by checking

table[index]!=NULL

1. Your completed program should display the contents of the hash table and the total number of collisions.

**Sample Outputs:**

**Text

Description automatically generated**

Text

Description automatically generated

**Final Code:**

**Node.h:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Node class declaration

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

class node

{ public:

node(int arg1=0,string arg2="",string arg3=""); // Constructor

void put(ostream &out); // Put data

private:

int id; // ID number

string first,last; // First and Last names

};

**Node.cpp:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Node class definitions

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;

#include "Node.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* node::set()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

node::node(int arg1,string arg2,string arg3)

{ id = arg1;

first = arg2;

last = arg3;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* node::put()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void node::put(ostream &out)

{ out << setw(8) << id;

out << setw(8) << first;

out << setw(16) << last;

}

**Main.cpp:**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Lab 28

\* Created by Twymun K. Safford

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <iomanip>

#include <string>

#include <fstream>

using namespace std;

#define TABLE\_MAX 21

#include "Node.h"

int hash\_insert(node\* table[], node\* p, int id);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* main()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

{

int i, id, count;

string fname, first, last;

fstream infile;

node\* table[TABLE\_MAX], \* p;

// Initialize

for (i = 0; i < TABLE\_MAX; i++)

table[i] = NULL;

count = 0;

// Open file

cout << "Enter file name: ";

cin >> fname;

infile.open(fname.data(), ios::in);

if (!infile.is\_open()) return;

// Read data from file

while (!infile.eof())

{

infile >> id >> first >> last;

// Process if input is valid

//assign p to values

//node\* p = new node(id, first, last);

p = new node(id, first, last);

if (infile.good())

{

//call the hash insert function

count += hash\_insert(table, p, id);

};

};

// Close file

infile.close();

// Display table

cout << endl;

for (i = 0; i < TABLE\_MAX; i++)

{

cout << left << setw(4) << i;

if (table[i] != NULL) table[i]->put(cout);

cout << endl;

};

// Collisions

cout << endl;

cout << "Collisions: " << count << endl;

// Clean up

for (i = 0; i < TABLE\_MAX; i++)

if (table[i] != NULL) delete table[i];

}

int hash\_insert(node\* table[], node\* p, int id)

{

//to store collisions

int collisions = 0;

//hash insert function to insert an element in side the table

//first compute the hashIndex

int index = (id \* 215 + 37) % 21;

//check if the index location is empty

if (table[index] == NULL)

{

table[index] = p;

}

//if location is not empty use linear hashing

//in linear hashing we move forward till we find an empty location where data can be inserted

else

{

int cur = index;

cur = (cur + 1) % TABLE\_MAX; //move cur forward

//move until back to same location as started

collisions = 1; //collison is 1 as already one collision has occured

while (cur != index)

{

//if empty location found insert the value

if (table[cur] == NULL)

{

table[cur] = p;

//return the number of collisions

return collisions;

}

else //if not an empty location increment cur and number of colisions

{

cur = (cur + 1) % TABLE\_MAX;

collisions += 1;

}

}

}

return collisions; //return final collisions if no location found

}