#include<iostream>

using namespace std;

const int Size=10;

class TelephoneLinear{

long table[Size] ;

public:

TelephoneLinear()

{

for(int i=0 ; i<Size ; i++)

table[i] = -1 ;

}

int hashFunction(long phone)

{

return phone % Size ;

}

void insert(long phone)

{

int index = hashFunction(phone);

int start = index;

while(table[index] != -1)

{

index = (index + 1) % Size ;

if(index == start)

{

cout<<"Hash table full\n";

}

}

table[index] = phone;

}

int search(long phone)

{

int index = hashFunction(phone);

int start = index;

int comparison = 1;

while(table[index] != -1)

{

if(table[index] == phone)

return comparison;

index = (index +1) % Size;

comparison++;

if(index == start)

break;

}

return -1;

}

void display()

{

for(int i=0 ; i<Size ; i++)

{

if(table[i] != -1)

cout<<i<<": "<<table[i]<<endl;

else

cout<<i<<": Empty\n";

}

}

};

class TelephoneQuadratic

{

long table[Size];

long phone;

public:

TelephoneQuadratic()

{

for(int i=0 ; i<Size ; i++)

{

table[i]= -1;

}

}

int hashFunction(long phone)

{

return phone % Size;

}

void insert(long phone)

{

int index = hashFunction(phone);

int i = 1;

int start = index;

while(table[index] != -1)

{

index = (start + i\*i) % Size;

i++;

if(i == Size)

{

cout<<"Hash table full\n";

return;

}

}

table[index] = phone;

}

int search(long phone)

{

int index = hashFunction(phone);

int i = 1;

int comparison = 1;

while(table[index] != -1)

{

if(table[index] == phone)

return comparison;

index = (index + i\*i) % Size;

i++;

comparison++;

if(i == Size)

break;

}

return -1;

}

void display()

{

for(int i=0 ; i<Size ; i++)

{

if(table[i] != -1)

cout<<i<<": "<<table[i]<<endl;

else

cout<<i<<": Empty\n";

}

}

};

int main()

{

TelephoneLinear l;

TelephoneQuadratic q;

long number[]={12313, 345, 567, 789};

int n = sizeof(number) / sizeof(number[0]);

cout<<"Inserting into linear probing\n";

for(int i=0 ; i<n ; i++)

{

l.insert(number[i]);

}

cout<<"Inserting into quadratic probing\n";

for(int i=0 ; i<n ; i++)

{

q.insert(number[i]);

}

cout<<"Linear probing table\n";

l.display();

cout<<"Quadratic probing table\n";

q.display();

cout<<"Search result\n";

for(int i=0 ; i<n ; i++)

{

cout<<"Searching for "<<number[i]<<endl;

int complinear = l.search(number[i]);

int compquadratic = q.search(number[i]);

cout<<"Linear probing comparison "<<complinear<<endl;

cout<<"Quadratic probing comparison "<<compquadratic<<endl;

cout<<endl;

}

return 0;

}