Practical-2: Implementation of hash functions and associated algorithms

//(Chaining)

```
include<iostream>
#include <list>
using namespace std;

class Hash
{
	int BUCKET;	list<int> *table;
public:
	Hash(int V);	void insertItem(int x);

		void deleteItem(int key);
	int hashFunction(int x) {
		return (x % BUCKET);
	}

	void displayHash();
};

Hash::Hash(int b)
{
	this->BUCKET = b;
	table = new list<int>[BUCKET];
}

void Hash::insertItem(int key)
{
	int index = hashFunction(key);
	table[index].push_back(key);
}

void Hash::deleteItem(int key)
{
int index = hashFunction(key);
list <int> :: iterator i;
for (i = table[index].begin();
		i != table[index].end(); i++) {
	if (*i == key)
	break;
}
if (i != table[index].end())
	table[index].erase(i);
}
void Hash::displayHash() {
for (int i = 0; i < BUCKET; i++) {
	cout << i;
	for (auto x : table[i])
	cout << " --> " << x;
	cout << endl;
```

```
}
}

int main()
{
int a[] = {15, 11, 27, 8, 12};
int n = sizeof(a)/sizeof(a[0]);
Hash h(7);
for (int i = 0; i < n; i++)
        h.insertItem(a[i]);

h.deleteItem(12);

h.displayHash();

return 0;
}
```

Output:

```
0
1 --> 15 --> 8
2
3
4 --> 11
5
6 --> 27
```