NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

Cachar, Assam

B.Tech. VIth Sem

Subject Code: CS-382

Subject Name: Introduction to Blockchain

Submitted By:

Name : Subhojit Ghimire

Sch. Id. : 1912160

Branch : CSE – B

ASSIGNMENT I: MERKLE TREE IMPLEMENTATION USING SELF IMPLEMENTED HASH FUNCTION.

THEORY:

The hash function used in this assignment resembles the most common and easiest hash function generation method called the Division Remainder. Division Remainder hash function can be described as:

$$h(k) = k \bmod n$$

where, 'k' is the key value

'n' is the size of the hash table.

The hash function is dependent upon the remainder of a division. It is best suited that 'n' is a prime number that can make sure the keys are more uniformly distributed.

In this assignment, the self-implemented hash function takes the input of a string and hashes it by traversing through the string. In each step, it adds up 1912 plus the multiplication of ASCII value of the character times 160. This is further passed under another mod operator which takes the mod 26 of the result and appends the xx^{th} character of the small case letter to the hash string, which is further updated with (result mod 10^4).

CODE:

```
#include<bits/stdc++.h>
using namespace std;

long mod = 10e4;

string hashFunction (string str) {
	unsigned long hash = 1;
	string ss = "";
	for (auto xx:str) {
		hash = hash + (1912 + (xx * 160));
		ss += char (97 + (hash % 26));
		hash %= mod;
	}
	return ss;
}
```

```
class merkleTree {
	private:
		deque<vector<string>> hashes;
		vector<string> transactions;

	public:
		void insertTransaction() {
			string inputString;
			cout << "ENTER STRING: ";
			cin >> inputString;

			int ii, zz, nn = hashes.size();
			transactions.push_back (inputString);
			if (nn == 0) {
				vector<string> vec;
				vec.push_back (hashFunction (inputString));
				hashes.push_back (vec);
				return;
			}

			hashes[0].push_back (hashFunction (inputString));
			for (ii = 1; ii < nn; ++ii) {
				zz = (hashes [ii - 1].size () / 2) + (hashes
[ii-1].size () % 2);
				if (hashes [ii].size() < zz)
					hashes	[ii].push_back	(hashFunction
(inputString));
				else {
					zz = hashes [ii].size() - 1;
					if ((zz << 1 | 1) < hashes [ii - 1].size
())
```

```
                                    hashes   [ii][zz]   =   hashFunction
(hashes [ii - 1][zz << 1] + hashes [ii - 1][zz << 1 | 1]);
                              else
                                    hashes [ii][zz] = hashes [ii - 1][zz
<< 1];
                        }
                  }

                  if (hashes [nn - 1].size () == 2) {
                        vector<string> vec;
                        vec.push_back (hashFunction (hashes [nn - 1][0]
+ hashes [nn-1][1]));
                        hashes.push_back (vec);
                  }
            }

            void printMerkleTree () {
                  if (transactions.size () == 0) {
                        cout << "UNDERFLOW: TREE EMPTY\n";
                        return;
                  }

                  cout << "\nMERKLE TREE: \n";
                  int ii, sz = hashes.size ();
                  for (ii = sz - 1; ii >= 0; --ii) {
                        for (auto yy:hashes [ii])
                              cout << yy << ' ';
                        cout << '\n';
                  }
            }

            void printRoot () {
                  int zz = hashes.size();
                  if (zz == 0)
```

```
                    cout << "UNDERFLOW: TREE EMPTY\n";
               else
                    cout << "\nROOT: " << hashes [zz - 1][0] <<
'\n';
          }

          void printTransactions () {
               if (transactions.size () == 0) {
                    cout << "UNDERFLOW: TREE EMPTY\n";
                    return;
               }

               cout << "TRANSACTION ARRAY: ";
               for (auto xx:transactions)
                    cout << xx << ' ';
               cout << '\n';
          }
};

int main () {
     merkleTree mrk;
     int option;
     system ("cls");
     do {
          cout << "\n#####################################\n";
          cout << "############### MENU ###############\n";
          cout << "#####################################\n";
          cout << "1. INSERT NEW TRANSACTION\n";
          cout << "2. DISPLAY HASHED ROOT\n";
          cout << "3. DISLAY TRANSACTIONS ARRAY\n";
          cout << "4. DISPLAY MERKLE TREE\n";
          cout << "5. EXIT\n";
          cout << "#####################################\n";
```

```
            cout << "ENTER YOUR CHOICE: ";
            cin >> option;
            while (option < 0 || option > 5) {
                  cout << "INVALID CHOICE! ENTER AGAIN: ";
                  cin >> option;
            }
            if (option == 1) mrk.insertTransaction ();
            else if (option==2) mrk.printRoot ();
            else if (option==3) mrk.printTransactions ();
            else if (option==4) mrk.printMerkleTree ();
            if (option >=2 && option <= 4) {
                  // cout << "\nPRESS ENTER TO CONTINUE\n";
                  system ("pause");
            }
      } while (option != 5);
      cout << "EXITING";
      return 0;
}
```

OUTPUT:

```
####################################
############### MENU ###############
####################################
1. INSERT NEW TRANSACTION
2. DISPLAY HASHED ROOT
3. DISLAY TRANSACTIONS ARRAY
4. DISPLAY MERKLE TREE
5. EXIT
####################################
ENTER YOUR CHOICE: 1
ENTER STRING: SUBHOJIT

####################################
############### MENU ###############
####################################
1. INSERT NEW TRANSACTION
2. DISPLAY HASHED ROOT
3. DISLAY TRANSACTIONS ARRAY
4. DISPLAY MERKLE TREE
5. EXIT
####################################
ENTER YOUR CHOICE: 1
ENTER STRING: GHIMIRE

####################################
############### MENU ###############
####################################
1. INSERT NEW TRANSACTION
2. DISPLAY HASHED ROOT
3. DISLAY TRANSACTIONS ARRAY
4. DISPLAY MERKLE TREE
5. EXIT
####################################
ENTER YOUR CHOICE: 2

ROOT: xfhvdzbzljjtjpl
Press any key to continue . . .
```

```
####################################
############### MENU ###############
####################################
1. INSERT NEW TRANSACTION
2. DISPLAY HASHED ROOT
3. DISLAY TRANSACTIONS ARRAY
4. DISPLAY MERKLE TREE
5. EXIT
####################################
ENTER YOUR CHOICE: 3
TRANSACTION ARRAY: SUBHOJIT GHIMIRE
Press any key to continue . . .

####################################
############### MENU ###############
####################################
1. INSERT NEW TRANSACTION
2. DISPLAY HASHED ROOT
3. DISLAY TRANSACTIONS ARRAY
4. DISPLAY MERKLE TREE
5. EXIT
####################################
ENTER YOUR CHOICE: 4

MERKLE TREE:
xfhvdzbzljjtjpl
jzrhzxrd ndxhbfj
Press any key to continue . . .

####################################
############### MENU ###############
####################################
1. INSERT NEW TRANSACTION
2. DISPLAY HASHED ROOT
3. DISLAY TRANSACTIONS ARRAY
4. DISPLAY MERKLE TREE
5. EXIT
####################################
ENTER YOUR CHOICE: 5
EXITING
PS D:\Documents\NITS\Semester VI\(THEOR
```