**Implementation of Shift-Add-XOR Hash Function:**

def hash\_function(key,length):

hashIndex=0

for ch in range(len(key)):

hashIndex=hashIndex ^ ((hashIndex << 5)+(hashIndex << 2)+ ord(key[ch]))

hashIndex=hashIndex % length

return hashIndex

**Implementation of hash Function:**

def hash\_function(key,length):

hashIndex=0

for ch in range(len(key)):

hashIndex=hashIndex+ord(key[ch])

hashIndex=hashIndex % length

return hashIndex

**Implementation of FNV hash Function:**

def hash\_function(key,length):

hashIndex=2166136261

for ch in range(len(key)):

hashIndex=(hashIndex \* 16777619) ^ ord(key[ch])

hashIndex=hashIndex % length

return hashIndex

**Implementation of ELF hash Function:**

def hash\_function(key,length):

hashIndex=0

for i in range(len(key)):

hashIndex=(hashIndex << 4) + ord(key[i])

g = hashIndex & 0xF0000000;

if (g != 0):

hashIndex = hashIndex ^ (g >> 24)

hashIndex = hashIndex & ~g;

hashIndex=hashIndex % length

return hashIndex

**Implementation of is\_prime functon in python.**

import math

def is\_prime(start\_val):

if(start\_val%2==0):

prime\_val=start\_val-1;

else:

prime\_val=start\_val;

isprime=False

while (not isprime):

prime\_val=prime\_val+2;

i=3;

isprime=True;

root=math.sqrt(prime\_val);

while(i<=root and isprime):

if(prime\_val%i==0):

isprime=False

else:

i=i+2;

return prime\_val;

**Basic Implementation of hash table:**

class hashTable:

def \_\_init\_\_(self):

self.size=10

self.arr=[None for i in range(self.size)]

def hashFunction(self,key):

h=0;

for i in key:

h=h+ord(i)

return h%10

def add(self,key,val):

k=self.hashFunction(key)

self.arr[k]=val

def get(self,key):

k=self.hashFunction(key)

return self.arr[k]

t=hashTable()

t.add('march 6',120)

t.add('march 12',130)

t.add('march 17',222)

print(t.arr)