MERKLE DAMGARD TRANSFORMATION CODE DOCUMENTATION

A Class named **merkle_damgard** is created incorporating all required variables and functions.

```
def __init__(self,message,pad_len):
    self.message = message
    self.pad_len = pad_len
    self.iv = generate_key(self.pad_len)
```

Here parameters are message, pad_len, and iv.
Message is the input message that needs to be hashed
Pad_length is the length for the fixed-length hash function

```
def pad_length(self):
    var = 0
    if len(self.message)%(self.pad_len):
        var = self.pad_len - len(self.message)%(self.pad_len)
    for i in range(var):
        self.message+="0"
    start = 0
    string_vec = []
    temp = ""
    for i in self.message:
        temp+=i
        start+=1
        if(start==self.pad_len):
            string_vec.append(temp)
            start = 0
            temp = ""
    print(string_vec)
    return string_vec
```

The function **pad_len** converts the message into the multiple of pad length by appending 0s at the end of the message.

After this the message is sliced into the strings with length equal to **pad_len** and stores them in an array named string_vec

```
def chain_prop(self):
    arr = self.pad_length()
    hash_func = fixed_len_hash(4,36389)
    hash_func.select_key()
    iv = self.iv
    for i in range(len(arr)):
        output = bin(hash_func.find_output(int(iv,2),int(arr[i],2)))[2:].zfill(self.pad_len)
        iv = output
    return output
```

Chain_prop function is the main part of the algorithm, a fixed length hash function is created and a key is generated, iv variable is initialized (it is an initial value).

After that in each iteration, an array element along with the output of the hash function of the previous iteration is put into the hash function and the loop continues.
The final hash value will be stored in the output variable.

```
if __name__=="__main__":
    hash_func = fixed_len_hash(4,36389)
    hash_func.select_key()
    # print("Hey ya")
    message = generate_key(100)
    print("message = ",message)
    print(len(bin(36389)[2:]))
    # val1 = random.randrange(0,36387)
    # val2 = random.randrange(0,36387)
    # print(len(bin(val1)[2:])+len(bin(val2)[2:]))
    # ans = hash_func.find_output(val1,val2)
    # print(len(bin(ans)[2:]))
    # print(hash_func.find_output(random.randrange(0,36387),random.randra
    merkle_damgard_hash = merkle_damgard(message,16)
    # merkle_damgard_hash.pad_length()
    print(merkle_damgard_hash.chain_prop())
```

The main part consists of all the classes as well as required variables and inputs.