Name: Shivraj Shankar Hulwan.

Emp Id: 2149291.

Section 1:

For storing data of card users we can use a Structure consisting of all the required parameters.

The Structure of card user’s is as follow.

Struct UserDetails{

Unsigned Long Social\_Security\_Number;

Unsigned Long Card\_Number; //Index

Unsigned Long CVV\_Number;

char DOB[10];

char Address[100];

char Full\_Name[50];

char Name\_Printed[20];

char Email[30];

char Card\_Expiry\_Date[10];

char Card\_Issue\_Date[10];

};

Size = 8+8+8+10+100+50+20+30+10+10 = 254 bytes + 2 bytes(padding) = 256 bytes

Hence for above Structure we require 256 bytes.

* Above Structure can be used to store the data of individual user.
* Here Card\_Number is used as Index as it is unique for everyone.
* For banks who want to use Name as Index, we can change our index to Full\_Name Array.
* For storing data of all users we can use a hashtable.
* We will create a hash function that will uniformly disturbute the data in the hashtable. In hashtable, every users data will be stored as a node of linked list.
* Hence Insertion and Search complexity of the data will be constant i.e O(1).
* For banks having persons name as index, we can modify our hash function. In this case we will process the ascii values of the user name and store the data acordingly.

Section 2:

In case of Banks having 15 digits, it can be saved in the same UserDetails structure as we are using Usigned Long for Card Number. But we need to change the hash function.

Section 3:

In case of memory constrained device we can omit Name\_Printed[] array and we can reduce the size of Address array and Full\_Name array and change its name to Name array. After reducing size Arrays are Address[50] and Name[25].