Week-4 UE20CS207 DSLAB

- Name : P K Navin Shrinivas
- SRN : PES2UG20CS237
- Section : D
- Batch : 2

Phone numbers with HashTable

Code :

main.c

```
#include "1_1.h"

int main() {
  struct person *table[tablesize] = {NULL}; // NULL reps EM
  while (true) {
    printf("1.Inset to Hash Table \n");
    printf("2.Search in Hash Table \n");
    printf("3.Delete from Hash Table \n");
    printf("4.Display entire Hash Table \n");
    int choice;
    printf("Enter Choice : ");
    scanf("%d", &choice);
    if (choice == 1) {
      struct person *d = (struct person *)malloc(sizeof(str
      printf("Enter phone number : ");
      scanf("%lli", &(d->phone));
      printf("Enter name : ");
      scanf("\t%[^\n]%*c", d->name);
      HashTableInsert(table, d);
    } else if (choice == 2) {
      printf("Enter phone number to search : ");
      ll ph;
      scanf("%lli", &ph);
      HashTableSearch(table, ph);
    } else if (choice == 3) {
      printf("Enter phone number to delete : ");
      ll ph;
      scanf("%lli", &ph);
      HashTableDelete(table, ph);
    } else if (choice == 4) {
      HashTableDisplay(table);
    } else {
      break;
    }
  }
}
```

1_1.h

```c
#include <math.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#define tablesize 1000
typedef long long int ll;

// to store the data into array
struct person {
  ll phone;
  char name[30];
};

void HashTableInsert(struct person **table, struct person *
void HashTableSearch(struct person **table, ll ph);
void HashTableDelete(struct person **table, ll ph);
void HashTableDisplay(struct person **table);
```

1_1.c

```
#include "1_1.h"
#include <stdio.h>

int HashFunction(ll key) {
  int fn = key % 10;
  key /= 10;
  fn += key % 10;
  int index = (fn * 2) % 7;
  return index;
}

void HashTableInsert(struct person **table, struct person *
  ll ph = d->phone;
  int index = HashFunction(ph);
  if (table[index] == NULL) {
    table[index] = d;
    printf("\n Inserted to Hash Table :) \n \n");
    return;
  } else {
    while (index < tablesize) {
      index += 1;
      if (table[index] == NULL) {
        table[index] = d;
        printf("\n Inserted to Hash Table UwU \n \n");
        return;
      }
    }
    printf("\n Table is FULL or right index not found for d
    return;
  }
}

void HashTableSearch(struct person **table, ll ph) {
  int index = HashFunction(ph);
  if (table[index] != NULL && table[index]->phone == ph) {
    printf("\n Record Found in Hash Table :) : \n ");
    printf("\n  Phone no : %lli ", table[index]->phone);
    printf("\n  Name : %s \n \n", table[index]->name);
    return;
  } else {
    while (index < tablesize) {
      index += 1;
      if (table[index] != NULL && table[index]->phone == ph
        printf("\n Record Found in Hash Table :) : \n \n");
        printf("\n  Phone no : %lli", table[index]->phone);
        printf("\n  Name : %s \n \n", table[index]->name);
        return;
      }
```

```
    }
    printf("\n Record Not found :( \n \n");
    return;
  }
}

void HashTableDelete(struct person **table, ll ph) {
  int index = HashFunction(ph);
  if (table[index]->phone == ph) {
    table[index] = NULL;
    printf("\n First record with that number deleted! \n \r
    return;
  } else {
    while (index < tablesize) {
      index += 1;
      if (table[index] != NULL && table[index]->phone == ph
        table[index] = NULL;
        printf("\n Firt record with that number deleted! \r
        return;
      }
      if (table[index] == NULL)
        index += 1;
    }
    printf("\n Record Not found :( \n \n");
    return;
  }
}

void HashTableDisplay(struct person **table) {
  int index = 0;
  printf("\n All records in the table : \n \n");
  while (index < tablesize) {
    if (table[index] != NULL) {
      printf("\n  Phone no : %lli", table[index]->phone);
      printf("\n  Name : %s \n \n", table[index]->name);
    }
    index += 1;
  }
}
```