#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

#define TABLE\_SIZE 10

struct node

{

char data[100];

struct node \*next;

}\*hashTable[26], \*curr;

//First Character

int hashFuction(char data[])

{

int hashKey = -1;

char firstChar;

//ambil karakter pertama

firstChar = data[0];

//A -> a (97) - 97 = 0;

hashKey = tolower(firstChar) - 97;

return hashKey;

}

//METHOD DIVISION

int divHash(int key)

{

return key % TABLE\_SIZE;

}

//METHOD FOLDING

int foldingHash(int key)

{

int part1 = key/1000; //ambil 4 digit pertama

int part2 = key%1000; //ambil 4 digit terakhir

return (part1 + part2) % TABLE\_SIZE;

}

void viewHashTable()

{

//access all index

for(int i=0; i<26; i++)

{

//cek index null atau tidak

if(hashTable[i] != NULL)

{

printf("Hash Table Index [%d] :\n", i);

curr = hashTable[i];

while(curr!=NULL)

{

printf("%s ", curr->data);

curr = curr->next;

}

printf("\n");

}

}

}

int rotatinghash(int key)

{

char keyStr[10];

sprintf(keyStr, "%d", key);

int len = strlen(keyStr);

if (len > 1)

{

// Rotasi: pindahkan digit pertama ke akhir

char rotatedStr[10];

strcpy(rotatedStr, keyStr + 1);

rotatedStr[len - 1] = keyStr[0];

rotatedStr[len] = '\0';

return atoi(rotatedStr) % TABLE\_SIZE;

}

return key;

}

void pushBack(int hashKey, char data[])

{

struct node \*newNode = (struct node\*)malloc(sizeof(struct node));

strcpy(newNode->data, data);

newNode->next = NULL;

//if the index is still empty

if(hashTable[hashKey] == NULL)

{

hashTable[hashKey] = newNode;

}

//if the index not empty

else

{

curr = hashTable[hashKey];

while(curr->next != NULL)

curr = curr->next;

curr->next = newNode;

}

}

int main()

{

char data[100];

int originalhash, rotatedhash;

do

{

printf("Input Data: ");

scanf("%s", data);getchar;

if(strcmpi(data,"exit")==0)break;

rotatedhash = rotatinghash(originalhash);

printf("Hash awal : %d, Rotated Hash: %d\n", originalhash, rotatedhash);

pushBack(rotatedhash, data);

} while(true);

viewHashTable();

getchar();

return 0;

}