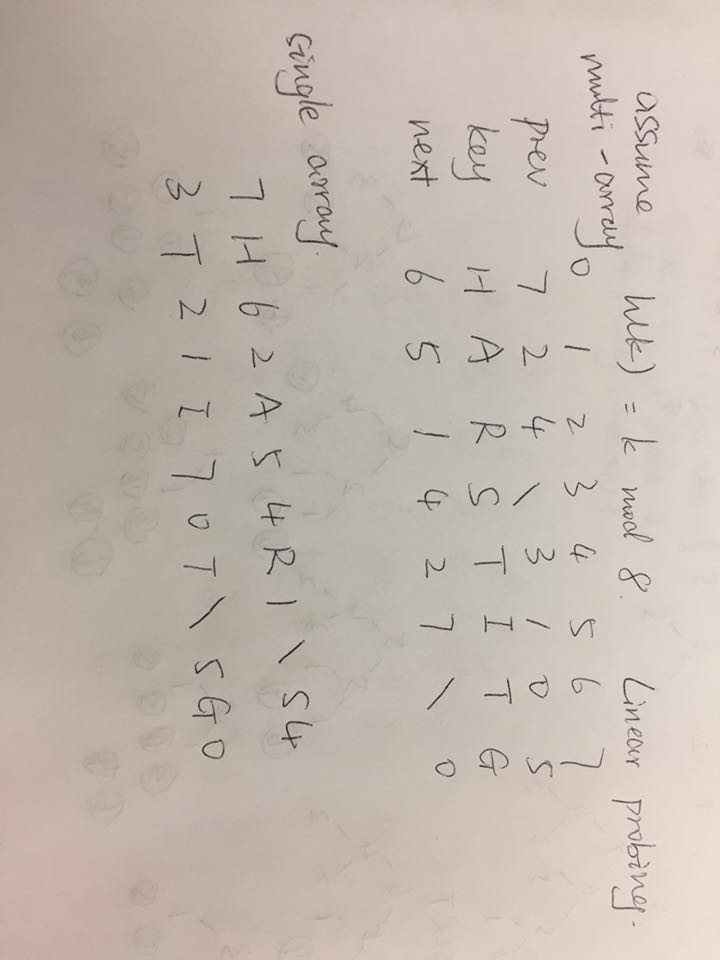
**CS 3050 Homework # 2. Name : Zhiqian Zhou**

**Submitted to Blackboard, due at 11:59pm on Feb. 15, 2018.**

1. Draw a picture of the sequence {S, T, R, A, I, G, H, T} stored as a doubly linked list using the multiple-array representation. Do the same for the single-array representation.



2. Give an efficient algorithm in pseudo code to reverse a singly linked list of n

elements. The algorithm should use no more than constant storage beyond that

needed for the list itself.

List-Reverse(L)

p <- head[L]

q <- next[p]

next[p] <- NIL

while(q != NIL)

r <- next[q]

next[q] <- p

p <- q

q <- r

head[L] <- p

3.Given an Hash function f(x) = x2 for 1000 integers in the range of –1000 to 1000, list 2 problems that could arises. Suggest a better Hash function to resolve these problems.

1. If there exist two integers whose abosolute value is the same, then there will be collisions.
2. According to the given range of integers, the Hash function will return the value ranged from 0 to 10^6. That’s totally a waste of space.

A = (sqrt(5) - 1) / 2, m = 2^11

h(k) = floor( m((k+1000)A mod 1))

4. Demonstrate what happens when we insert the keys 3, 11, 80, 74, 92, 1024, 32, 59, 503, 293, 2010, 22, 104 into a hash table with collisions resolved by chaining. Let the table have 11 slots, and let the hash function be h(k) = k mod 11.

// hash.c

#include <stdlib.h>

#include <stdio.h>

typedef struct \_node{

int val;

struct \_node \*next;

}node;

node\* hash[11];

int h(int k){

return k % 11;

}

void insert(int val){

int index = h(val);

node\* cur = hash[index];

if(cur == NULL){

hash[index] = (node\*)malloc(sizeof(node));

cur = hash[index];

}

else{

while(cur->next != NULL) cur = cur->next;

cur->next = (node\*)malloc(sizeof(node));

cur = cur->next;

}

cur->val = val;

cur->next = NULL;

}

void print(){

int i = 0;

for(; i < 11; ++i){

printf("i = %d: ",i);

if(hash[i] == NULL) printf("\n");

else{

node \* cur = hash[i];

while(cur != NULL){

printf("%d ",cur->val);

cur = cur->next;

}

printf("\n");

}

}

printf("\n");

}

int main(){

int array[] = {3, 11, 80, 74, 92, 1024, 32, 59, 503, 293, 2010, 22, 104};

int len = sizeof(array)/sizeof(array[0]), i = 0;

memset(hash,NULL,sizeof(hash));

for(; i < len; ++i){

printf("insert:%d\n",array[i]);

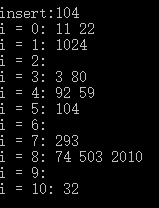
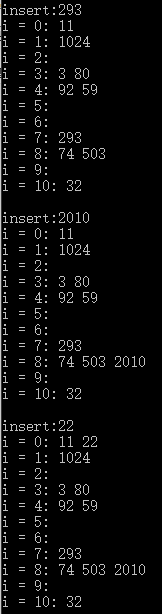
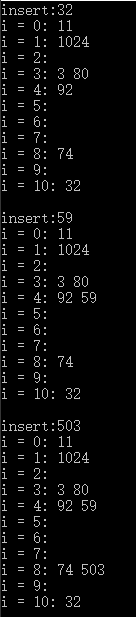
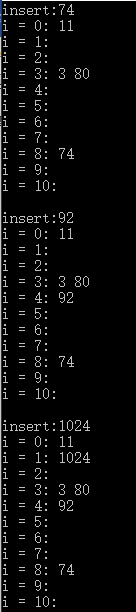
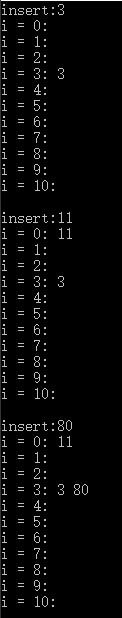
insert(array[i]);

print();

}

return 0;

}



5. Consider a hash table of size m = 128 and a corresponding hash function h(k) = floor( m (kA mod 1) ) for A = (sqrt(5)-1)/2. Compute the locations to which the keys 1000, 1001, 1002, 1003, and 1004 are mapped.

// hash.c

#include <stdlib.h>

#include <stdio.h>

#include <math.h>

double A = (sqrt(5)-1) / 2.0, m = 128;

double h(int k){

return floor(m \* fmod(k\*A, 1.0));

}

int main(){

printf("h(1000) = %.0f\n",h(1000));

printf("h(1001) = %.0f\n",h(1001));

printf("h(1002) = %.0f\n",h(1002));

printf("h(1003) = %.0f\n",h(1003));

return 0;

}

