WEEK 1

Wednesday 20th.

Virtual Machine Software

* Vm players from vmware.com
* Virtualbox.org as well
* Newest ubantu
  + 2 gig memory and 2 processor

File is linkedList.h

#ifndef LINKEDLIST\_H

#define LINKEDLIST\_H

#include<stdio.h>

#includestdlib.h>

Struct node {

Void \* data;

Struct node next;

};

Typedef struct node Node;

Struct linkedList {

Node \* head;

Int size;

};

Typedef struct linkedList LinkedList;

Linkedlist \* linkedlist();

LinkedList.c

LinkedList \* linkedlist() {

LinkedList \* mylist = ((linkedList\*) calloc(1,sizeof(LinkedList));

Malloc(1 \*sizeof((linkedList));

MyList ->head = NULL;

MyList->size = 0;

MyList -> head = (Node \*) calloc(1,sizeof(node));

MyList->head->next = NULL;

}

Void \* buildInt(FILE \*fin) {

Char temp[100];

Fgets(temp, 100, fin);

Int \* val = (int \*) calloc(1,sizeof(int));

(\*val) = atoi (temp);

Return val;

}

Node \* buildNode(void\* (\*buildtype)(FILE \*fin)) {

Node \*nn (Node \* calloc(1,sizeof(node));

Nn->data = buildtype(fin);

Return nn;

}

**Thursday**

Void cleanTypeInt(void\*ptr) {

Int \* temp = (int \*) ptr

Free(temp);

Temp = Null;

}

Void \* build typeInt (FILE \* fin) {

Char temp[MAX];

Int \* aint = (int \*) calloc (1, size of (int));

Fgets(temp, 100, fin);

\*aint = atoi (temp);

Return aInt;

}

Void printTypeInt(void \* passedIn) {

Int temp = \*(int\*)passedin);

Pf(“val %d \n”, temp)

}

Int compareInt(const void \* p1, const void \*p2) {

Int left = \*((int \*) p1);

Int right = \*((int \*) p2);

Return left – right;

}

LinkedList \* linkedlist() {

LinkedList \* myList = (linkedList \*) calloc (1, sizeof(LinkedList));

Return my list;

}

Void printList(LnikedList \* myList, void(\*convert data)(void \*ptr)) {

If (mylist->size == 0) {

Pf(“empty”)

} else {

Node \* cur = myList ->head)

While(cur !=NULL) {

ConvertData(cur->data);

Cur = cur-> next;

}

}

Void addFirst(LinkedList \* myList, Node \* nn) {

nn->next = mylist ->head;

myList ->head = nn;

myList ->size = myList ->size +1;

}

Void removeFirst(Linkedlist \* thelist, void(\*removeData)(void\*)) {

If\*thelist -> size == 0) {

Pf empty

}

Node \* n = TheList -> head;

TheList -> head = theList->head->next;

TheList ->size = theList->size – 1;

RemoveData(n ->data);

Free(n)

N = NULL;

}

**FRIDAY**

Memory Map

Gcc main.c

-preprocess

-compiler ->.i

-assembler .i->.s

Static linker runs -> .o ROF

-ld brings in cstdlib -> ./a.out -> elf

Free data passed in.

Void clearList(LinkedList \* theList, void(\*remove Data)(void\*)) {

If(the list size == 0) {

Node \* cur = theList->head;

Node \* temp;

}

While(cur not Null) {

TEMP = CUR;

Cur – cur.next;

Remove data(temp.data)

Temp.data = NULL;

Free(temp);

}

TheList ->head = null;

Thelist size = 0;

}

Void Sort(Linkedlist\* theList, int(\*compare)(const void\*, const void\*)) {

Node \* start, \*search, \*min;

Void \* temp;

If(theList -> size > 1) {

For(start = the list -> head, start -> next != NULL; start =start ->next) {

Min = start;

For(search = start -> next; search !=NULL; search = SEARCH.NEXT) {

If(compare (search.data, min.data) < 0)

Min = search;

}

Temp = start -> data;

Start -> data = min -> data;

Min -> data = temp;

}

}

}