**Learn C the hard way**

Part 2

**Linked List**

A Linked list works by nodes having pointers to their next or previous element. A doubly linked list contains pointers to both while a singly linked list only points at the next element.

Anything that involves inserting or deleting an element will be very fast. The main disadvantage is that traversing a linked list involves processing every single pointer along the way, in which case sorting and searching will be slow.

Example – linked list

Typedef struct Node {

Int data;

Struct Node \*next;

} Node;

Int main() {

Node \*head = NULL;

Node \*second = NULL;

Node \*third = NULL;

Head = malloc(sizeof(Node));

Second = malloc(sizeof(Node));

Third = malloc(sizeof(Node));

Head->data = 1;

Head->next = second;

Second->Data = 2;

Second->next = third;

Fprintf(stdout, “%d\n”, head->next->data);

Return 0;

}

**Hash map**

Hash maps (aka hashmaps, hashes, dictionaries) are used for storing key/value data. A hash map works by performing a hashing calculation on the keys to produce an integer, then use that integer to find a *bucket* to get or set the value. A JavaScript object is an example of a hash map.

// ***NEED EXAMPLE!!!!!!!!!!!!!!!!!!!!!***

**Stacks & Queues**

Stacks & queues are simple data structures that are really variants of the *List* data structure. All they do is use a list with a discipline that says *you always place elements on one end of the list*.

A stack you always push and pop – LIFO

A queue you always shift to the front but pop from the end (FIFO)

Example – stack

Struct Stack {

Int top;

Unsigned capacity;

Int \*array;

};

Struct Stack \*createStack(unsigned capacity) {

Struct Stack \*stack = malloc(sizeof(struct Stack));

Stack->capacity = capacity;

Stack->top = -1;

Stack->array = malloc(stack->capacity \* sizeof(int));

Return stack;

}

Int isFull(struct Stack \*stack) {

Return stack->top == stack->capacity – 1;

}

Int isEmpty(struct Stack \*stack) {

Return stack->top == -1;

}

Void push(struct Stack \*stack, int item) {

If (isFull(stack)) {

Return;

}

Stack->array[++stack->top] = item;

Fprintf(stdout, “%d pushed to stack\n”, item);

}

Int pop(struct Stack \*stack) {

If (isEmpty(stack)) {

Return INT\_MIN; // from <limits.h>

}

Return stack->array[stack->top--];

}

Int peek(struct Stack \*stack) {

If (isEmpty(stack)) {

Return INT\_MIN;

}

Return stack->array[stack->top];

}

Int main() {

Struct Stack \*stack = createStack(100);

Push(stack, 100);

Push(stack, 200);

Push(stack, 500);

Fprintf(stdout, “%d popped from stack\n”, pop(stack));

Return 0;

}

**QUIZ**

1. Add push, pop and unshift functions to linked list
2. Create a queue
3. Add function to stack which will print out the full list