CMPT 125 Assignment 6

February 22, 2019

- Due at 15:20:00 on Wednesday, Feb. 27
- Please make your submission on CourSys by uploading the following:
  - A .c file containing your code for Question 1
  - A .c file containing your code for Question 2
  - (No .pdf file is required for this assignment)
- Please include comments to explain all your work
- Remember to consider the typical case as well as any appropriate corner cases

Question 1 (25 Marks)

In class, we have studied the singly linked list, implemented as follows: (Code can be found in the accompanying .c file)

```
#include <stdio.h>
#include <stdlib.h>

typedef struct _node {
    int data;
    struct _node * next;
} node_t;

typedef struct {
    node_t * head;
    node_t * tail;
} LL_t;

LL_t * LLcreate() {
    LL_t * ret = malloc(sizeof(LL_t));
    ret->head = NULL;
    ret->tail = NULL;
    return ret;
}

void LLappend(LL_t * intlist, int value) {
    node_t * newNode = malloc(sizeof(node_t));
```

```
    newNode->data = value;
    newNode->next = NULL;

    if (intlist->head == NULL) {
        intlist->head = newNode;
        intlist->tail = newNode;
    } else {
        intlist->tail->next = newNode;
        intlist->tail = newNode;
    }
}
```

Part a)

Implement a function "LLMax" to return the biggest number in the linked list. For example, if a linked list has the elements {5, 100, -100, 2019}, the function LLMax should return 2019. If the linked list is empty, the function should return any integer of your choice, and display the message "Empty List!" The function prototype is given below.

```
// Returns the biggest number in the LL_t
int LLMax(LL_t * intlist) {
}
```

Part b)

Implement a function "LLDelete" which removes all occurrences of a target number if it is found in the linked list, and displays "Value not found" if the target number is not found. The function prototype is given below.

```
// Deletes the node containing the target integer, and
// warns user if the target is not found
void LLDelete(LL_t * intlist, int target) {
}
```

Question 2 (25 Marks)

In this question, we will learn about a variant of linked list called "doubly Linked List". In addition to the "next" pointer pointing to the next node in the list, a node in a doubly linked list also contains a pointer "prev" pointing to the previous node in the list.

One implementation of the doubly linked List is as follows. (Code can be found in the accompanying .c file)

```c
#include <stdio.h>
#include <stdlib.h>

// Node definition for doubly linked list
typedef struct _dllnode  {
    int data;
    struct _dllnode * next;
    struct _dllnode * prev;
} DLLnode_t;

// Definition of doubly linked list
typedef struct {
    DLLnode_t * head;
} DLL_t;

// Creates a doubly linked list
DLL_t * DLLCreate() {
    DLL_t * ret = malloc(sizeof(DLL_t));
    ret->head = NULL;
    return ret;
}

// Appends a DLLnode_t containing the value x into a DLL_t
void DLLAppend(DLL_t * intlist, int x) {
    // Create a DLLnode_t
    DLLnode_t * newNode = malloc(sizeof(DLLnode_t));
    newNode->data = x;
    newNode->prev = NULL;
    newNode->next = NULL;

    // Point head to new node if list is empty
    if(intlist->head == NULL) {
        intlist->head = newNode;
        return;
    }

    DLLnode_t * temp =intlist->head;
    while(temp->next != NULL) {
        temp = temp->next; // Go To last Node
    }

    temp->next = newNode;
    newNode->prev = temp;
}

// Prints the elements of a doubly linked list
void DLLPrint(DLL_t * intlist) {
    DLLnode_t * temp = intlist->head;
    while(temp != NULL) {
        printf("%d ",temp->data);
        temp = temp->next;
    }
    printf("\n");
}
```

Part a)

Based on the doubly linked list implementation given above, implement a function called "DLLReverse" to reverse a list. For example, if a doubly linked list originally contains the list {4, 5, 7, 7, 9}, after calling DLLReverse, the list would become {9, 7, 7, 5, 4}. The function prototype is given below.

```
// Reverses a DLL_t
void DLLReverse(DLL_t * intlist) {
}
```

Part b)

Implement a function called "DLLRemove" to remove any node at a specified index "ind". Just as with C++ arrays, the first element of the list should have an index of 0. Starting at 0, indices go up to one less than the length of the list. Your DLLRemove function should give the following message if the user passes in an invalid index: "Warning: Invalid index!" No other operations should be done if the index is invalid. The function prototype is given below.

```
// Removes the element at index ind for a DLL_t, and
// warns the user if the index is invalid
void DLLRemove(DLL_t * intlist, int ind) {
}
```