CS 3423.002

Assignment 3: Implementing a doubly linked list using C language

Due: Midnight on 12:05am of 10/27/2014

1 Overall Description:

Linked list is one of the fundamental data structures in C. In this assignment, you will implement a doubly link list by yourself. Description about doubly linked list can be seen from http://en.wikipedia.org/wiki/Linked_list#Doubly_linked_list.
The structure and function declaration can be seen in the following, which is the dllist.h.

```
typedef int valueT;
typedef int bool;

typedef struct list {
  valueT        value;
  struct list * prev; // Pointing to previous node
  struct list * next; // Pointing to next node
} list_t;

// Initialize an actual link list (list).
void dl_init(list_t * list);

// Insert a value (val) into the tail of the given list (list).
void dl_insert(list_t * list, valueT val);

// Delete the node with give value (val) from the given list (list).
bool dl_del(list_t * list, valueT val);

// Insert a value (val) into the front of the given list (list), which becomes the first node o
void dl_insert_front(list_t * list, valueT val);


// Get the first node's value from the given list (list).
valueT dl_get_front(list_t * list);

// Get the last node's value from the given list (list).
valueT dl_get_back(list_t * list);

// Delete the first node from the given list (list).
void dl_pop_front(list_t * list);

// Delete the last node from the given list (list here).
void dl_pop_back(list_t * list);

// Check whether the value (val) is existing or not. If existing, it is return 1. Otherwise,
// this function will return 0.
bool dl_check(list_t * list, valueT val);
```

NOTE:Those parameters and names of these functions should be exactly the same as listed here since I will write my test programs to verify these functions that your implemented.

2 Submission Requirements

Here is the description about what you should submit in the end. Points for the program will be 60% and 40% will be the readme file. I expect around two-pages writeup for this assignment.

- dllist.c: The actual implementation of those functions.
- dllist.h: list_t structure's definition and those functions.
- test.c: You are putting all test cases that you have designed here.
- readme.txt: A simple text file, not a pdf file. See below for the descriptions.

2.1 readme

In this part, you will describe the following things:

- Function Implementation Description: You will simply describe how to design every listed functions. For example, dl_get_front will check whether the list is empty or not. If it is not empty, then this function will return the node pointing by "next" pointer.
- Building and Using as a shared library: You should describe the procedure of compiling these files as a dynamic library. How you should compile your program? Putting actual command line here.
- Building a static library: You should describe the procedure of compiling these files as a static library. How you should compile your program? Putting actual command line here.
- Comparing between dynamic library and static library: You should list the size information of different types of library. Also, you will compare the difference of using these two different libraries. What is the advantage of using static library and using dynamic library? Similar to what we discussed in the class.
- Designed Test Cases: Describe those test cases that you have designed (and in your test.c file) for evaluating those test cases. I will give points for the designe of test cases designs. More coverage will be considered to be better. You are expected to have at least one or two test cases for each function.

2.2 Suggested Tests:

You designed these by yourself.