CMPT125, Fall 2018

Homework Assignment 3
Due date: November 2, 2018

Submit homework, printed or written in readable handwriting,
to the assignment boxes in CSIL ASB9838.

1) [15 points] Stacks:
In this question you may assume that the functions below are implemented.
You cannot make assumptions about how the stack is implemented.
If you want to use any other function, you need to implement it.

```
typedef struct {
   ... // not known
} stack_t;

// creates a new stack
stack_t* stack_create();

// pushes a given item to the stack
void stack_push(stack_t* s, int item);

// pops the top element from the stack
// Pre condition: stack is not empty
int stack_pop(stack_t* s);

// checks if the stack is empty
bool stack_is_empty(stack_t* s);

// frees the stack
void stack_free(stack_t* s);
```

[12 points] Write a function that gets two stack and checks if they are equal (i.e., have the same elements in the same order). When the function returns the stacks must be in their initial state.

```
// checks if the two stacks are equal
bool stack_equal(stack_t* s1, stack_t* s2);
```

[3 points] What is the running time of your function?

2) [15 points] Queues:
In this question you may assume that the functions below are implemented.
You cannot make assumptions about how the queue is implemented.
If you want to use any other function, you need to implement it.

```
// creates a new queue
queue_t* queue_create();

// enqueue a given item to the queue
void enqueue(queue_t* q, int item);

// remove from the queue, and return the removed value
// Pre condition: queue is not empty
int dequeue(queue_t* q);

// checks if the queue is empty
bool queue_is_empty(queue_t* q);

// frees the queue
void queue_free(queue_t* q);
```

[12 points] Write a function that creates a copy of a queue, i.e., it gets a queue and creates another queue with the same elements in the same order.
In the end on the function, the original queue must be returned to its initial state.

```
// returns a copy of orig
queue_t* queue_copy(queue_t* orig);
```

[3 points] What is the running time of your function?

3) [38 points] Linked List:
You may only use the struct and the functions below.
If you want to use any other function, you need to implement it.

```
// A node in linked list
struct node {
  int data;
  struct node* next;
};
typedef struct node  node_t;

// Linked list
typedef struct {
  node_t* head; // pointer to the head of the list
} LL_t;

// Adds a new element to the head of a list
void LL_add_to_head(LL_t* list, int value);

// Removes the first element from a list
int LL_remove_from_head(LL_t* list);

// Returns true if the list is empty, and returns false otherwise
bool LL_is empty(LL_t* list);

// frees the linked list
void LL_free(LL_t* list);
```

(a) **[15 points]** Write a function that gets a linked list, and reverses it.
For example, if the input is 1→ 2→ 3→ 4, then after applying the function, the list will be 4→ 3→ 2→ 1.

```
void LL_reverse(LL_t* list);
```

[4 points] What is the running time of your function?

(b) **[15 points]** Write a function that gets a linked list and a parameter k, and removes the k'th element from the end of the list. (For k=0, we need to remove the last element).
For example, if the input list is 1→ 2→ 3→ 4→ 5, and k=2, then after applying the function, the list will be 1→ 2→ 4→ 5.
If k >= length of the list, then the list remains the same

```
void LL_remove_kth_from_end(LL_t* list, int k);
```

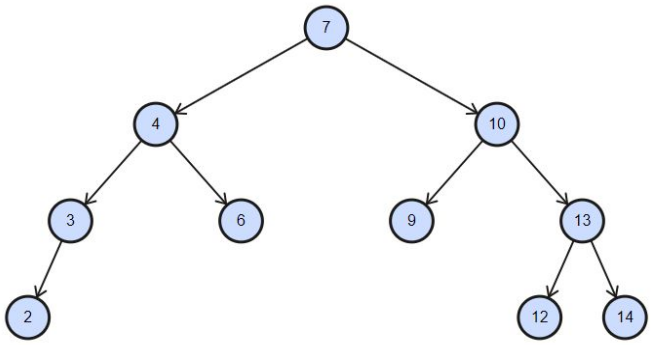
[4 points] What is the running time of your function?

4) [35 points] Binary Trees:
In this question you may only use the struct and the functions below.
If you want to use any other function, you need to implement it.

```
// Binary Tree node
struct BTnode {
 int data;
 struct BTnode* left;
 struct BTnode* right;
 struct BTnode* parent;
};
typedef struct BTnode BTnode_t;


// creates a BTnode_t data=value, and all pointers set to NULL
BTnode_t* create_node(int value);
```

(a) **[12 points]** What are the Inorder, Preorder and Postorder traversals of the binary tree below.



(b) **[8 points]** Draw two different trees T_1 and T_2, with all values in T_1 distinct, such that preOrder(T_1)=preOrder(T_2) and postOrder(T_1)=postOrder(T_2) .

(c) **[10 points]** Write an algorithm that gets two sequence of distinct ints (no repetitions), representing the inorder and the preorder traversals of some binary tree, and returns the tree.
[5 points] Prove that for a given inOrder and preOrder, the tree is unique.

For example, if inOrder=[1,2,3,4] and preOrder=[3,2,1,4], then the unique binary tree is