AE1PGA Lab 8

Expanding Integer List

This task is to create functions to handle single-linked lists. The file with the name `int_list.c(incomplete)` contains the definition of a structure which can hold an integer value and a pointer to the next list structure in the list and a few functions, to append, print the list. Based on the lecture, complete the implementation of such a data-structure with the following functions. In each case, think about what different situations you may have (eg, empty vs non-empty list) and how you could handle errors.

Textbook Chapter 12 also covers data-structures in C and might be helpful (be careful of the type definitions though. The book likes to `typedef` pointer types, which I think is confusing because it can make things that are pointers look like they are not, and vice-versa).

- `list_prepend` that takes a pointer to a pointer to the first element of a list and a new value, and adds that new value to the start of the list.
  `int list_prepend(IntList **start, int newval);`
- `list_insert` that takes a pointer to a pointer to the first element of a list, a new value, and an index, and adds that new value before the n'th element of the list.
  `int list_insert(IntList **start, int newval, int index);`
- `list_get` that takes a pointer to the first element of a list and an index, and returns the the value at that index.
  `int list_get(IntList *start, int index);`
- `list_contains` that takes a pointer to the first element of a list and a target value, and returns true or false if the list does or does not contain that value.
  `int list_contains(IntList *start, int target);`
- `list_remove` that takes a pointer to a pointer to the first element of a list and an index, and removes that element at that index from the list.
  `void list_remove(IntList **start, int index);`
- `list_free` that takes a pointer to the first element of a list, and free all allocated memory.
  `void list_free(IntList *start);`

End