Jordan Ditzler – G00967092
ECE 446 – FALL 2020

Homework 2

Structures included:

```
struct node
{
    struct node* prev;
    struct node* next;
    int bib, time;
    char *name, *school;
};
```

This is the node structure that makes up my linked list. It contains a pointer to the next and previous node in the list as well as containing the four required data variables.

Functions included:

- `void insert_entry(struct node** head, int *bib, int *time, char *name, char *school);`

  Function:

  > Inserts a new node at the head of the current list, becoming the new head node. Allocates memory for the new node and inserts the data passed to the function into the node.

  Parameters:

  > `struct node** head`: Double pointer to the head node of which to add to.
  > `int *bib`: Pointer of bib data variable to be added to the node.
  > `int *time`: Pointer of time data variable to be added to the node.
  > `char *name`: Pointer of name string data variable to be added to the node.
  > `char *school`: Pointer of school string data variable to be added to the node.

- `void remove_entry(struct node** head, struct node* remove);`

  Function:

  > Removes a node from the head of the list, making the next-in-line node the new head. Frees the memory associated with the deleted node.

  Parameters:

  > `struct node** head`: Double pointer to the head node of which to delete from.
  > `struct node* remove`: Pointer to the head node of which to delete.

- `void print_list(struct node** head);`

  Function:

  > Prints out all the data in each node, going from head to tail.

  Parameters:

  > `struct node** head`: Double pointer to the head node of which to start printing from.

Testing:

Simply run the code after compilation (steps included in Readme file) to start testing. The test is a simple, iterative process which adds 8 nodes to a list, prints out the list, removes the 8 nodes from the list, and starts again. The test also has checks in between steps to see if the list empty or populated to make sure the program is properly adding and deleting nodes as well as keeping the head pointer updated. All significant steps in the tester will have an associated printout in the terminal that details the step that has taken place. Default test iterations is 2, but more can be done via editing the `i < 3` in line **123** of **hw2.c**. The test has been confirmed to work for iterations up to 1000000.