SP(H) - Lab Sheet #3

Yehia Elkhatib

The topics for this lab sessions are:

- pointers
- memory allocation

Task 3.A

► Write a function that implements the following prototype: `void reverse_print(char **p, int n);`
The function takes an array of strings **p** and a number of words **n**, then prints the first **n** strings in reverse order.
► In the main function, you will need to declare the message to be reversed as such:
`char *message[WORDS_MAX] = {"I", "think", "we've", "got", "our", "roles", "reversed"};`
where `WORDS_MAX` is a constant defined using the `#define` directive. Set its value to 50.
► Using the above declaration, calling `reverse_print(message, 3);` should result in the output:
we've think I;
while calling `reverse_print(message, WORDS_MAX);` should print: **reversed roles our got we've think I**.
► Make sure your code is robust enough for when the size of the actual message is shorter than the maximum size it could be. Test this using a large value of `WORDS_MAX` (e.g. 1,000).

Task 3.B

► On Moodle you are provided with a file that partially implements a basic linked list structure. It also includes a driver, i.e. main method that could be used to test that your code works.
► You are asked to write implementation for the 2 following functions:

1. `create_node` attempts to allocate a new node using `malloc`. If it fails, it returns NULL. Otherwise, it sets the members (value and next) to appropriate values.
2. `free_list` frees the whole list starting from the node.