CSci 2113 - Lecture 4: C Structs and Strings

Prof. T. Wood   Fall 2016

1. What is wrong with the code below? Fix it. Then fill in the stack and heap memory of the program right before it exits. Assume `ints` and pointers uses 4 bytes and no extra space is needed for function headers on the stack, or for heap meta data. The stack starts at 10,000 and the Heap starts at 50,000, both grow upwards.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct car {
  char make[50];
  char model[50];
  int year;
};

int main(){
  struct car c1;
  struct car* c2;
  // Only change code below this point

  strcpy(c1.make, "Honda");
  strcpy(c1.model, "Fit");
  c1.year = 2015;


  strcpy(c2->make, "Fiat");
  strcpy(c2->model, "500");
  c2->year = 2016;
  return 0;
}
```

Stack

Address	*Name*	*Contents*
10000		

Heap

Address	*Allocated?*	*Contents*
50000		

2. Write a helper function in
`lec-4/clean_newline.c`
that takes a string and removes the \n at the end.

To compile the code, run:
`make`
To run the program, run:
`./cleaner`

3. Open the code in `lec-4/romeo/rj.c`

`a)` Modify the code so that it updates **`num_chars`** with the total length of strings read from the file. Have it print out both the total size of the lines[][] array and the total length of the strings.

`b)` This is clearly wasteful. Modify the current program logic below so that it allocates the appropriate amount of memory needed for each line read in from the file.

```
create a 2D array with plenty of space for each line

open the text file for reading

for the first 20 lines of the file...

    read a line from the file into a temp variable

    if there was an error, break out of for loop

    copy the temporary line into the next entry in the array
```

!!!!Do not proceed until an instructor approves your modified algorithm!!!!

`c)` Modify the code to match your new algorithm.