CpSc 2120: Algorithms and Data Structures

Instructor: Dr. Brian Dean — Fall 2018
Webpage: `http://www.cs.clemson.edu/~bcdean/` — TTh 12:30-1:45
Handout 8: Quiz 1 Coding Part. 25 possible points. — Earle 100

You may use any resources associated with this class for this quiz (e.g., pre-existing notes or code you have written for this class, code or notes from the class Canvas site). You can look at man pages and `cppreference.com` for general C/C++ reference, but you may not consult any other material from the web. Feel welcome to use prior code from the Canvas site in your solutions if you feel this is helpful.

1 Riffle Shuffles (5+5 Points)

Given two stacks A and B of playing cards, a *riffle shuffle* combines them into a single stack[1] by interleaving them randomly. The first card of the combined stack is equally likely to be the first card of A or the first card of B, and each subsequent card in the combined stack is randomly taken from A or B, again choosing with equal probability between these options. The cards from A and B therefore stay in the same order during the process.

Please write two functions that perform riffle shuffles, the first using arrays and the second using linked lists. The first function should have this prototype:

```
int *riffle(int *A, int *B, int n);
```

It takes pointers to two integer arrays A and B, each of length n, as input, and returns a pointer to an array of length $2n$ (dynamically allocated by your function) whose contents is a riffle shuffle of A and B.

The linked list version should have this prototype:

```
struct Node {
  int val;
  Node *next;
}

Node *riffle(Node *A, Node *B);
```

Given a pointer to the head nodes of two linked lists of card values, it should return a pointer to a *newly allocated* linked list representing a riffle shuffle of A and B. The original linked lists are not to be modified. You may add constructors to the definition of the `Node` structure if you like. Your code should work for lists of length up to 10 million nodes.

[1]The word "stack" here does *not* refer to the "stack" data structure; it just means a pile of cards in some order.

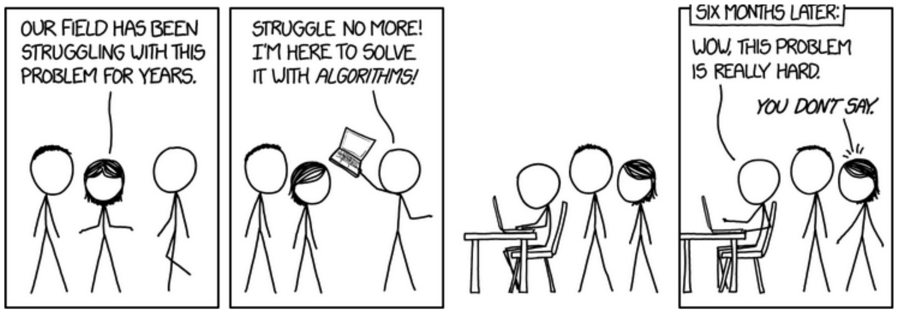
OUR FIELD HAS BEEN STRUGGLING WITH THIS PROBLEM FOR YEARS.
STRUGGLE NO MORE! I'M HERE TO SOLVE IT WITH *ALGORITHMS!*
SIX MONTHS LATER:
WOW, THIS PROBLEM IS REALLY HARD.
YOU DON'T SAY.

Figure 1: An example of an xkcd comic, by Randall Munroe.

2 Explaining Things (15 points)

Randall Munroe, author of the popular *xkcd* web comic (a favorite of nerds everywhere), has recently published a clever book entitled *Thing Explainer*, in which he attempts to explain complicated things but only using the 1000 most common words in the English language. Examples include "Sky boat with turning wings" (a helicopter) and "Boxes that makes clothes smell better" (a washing machine and dryer).

As you know, Prof. Dean is also writing a book (on algorithms), and he is wondering if he is using simple language like Randall Munroe's book or not. In the directory

`/group/course/cpsc212/f18/hw01/`

you will find two new files: `e.txt`, which contains around 4 million words taken from news stories in 2016, and `s.txt`, which contains the sequence of words from the sorting chapter of Prof. Dean's book. Regarding the contents of `e.txt` as being representative of the English language, please determine the 1000 most common words in `e.txt` (don't worry, there are no ties between the 1000th and 1001st most common words), and then print out all the words in `s.txt` that are *not* in this set (i.e., print out all the words in Prof. Dean's chapter that are uncommon). You should print each word at most once, even if it occurs in `s.txt` multiple times. Please print out all the uncommon words in order of obscurity – i.e., with the most common ones first, ending with the least common ones. Words with the same level of obscurity can be printed in any order.

Note that there are two buckets on handin to which you should submit code for this midterm – one for the first question and one for the second question. Note also that partial credit will be given to solutions that make progress towards the final goal but don't necessarily accomplish every objective, particularly for the second problem.