CS3520 – Programming in C++
Fall 2015
Assignment 3

Assignment

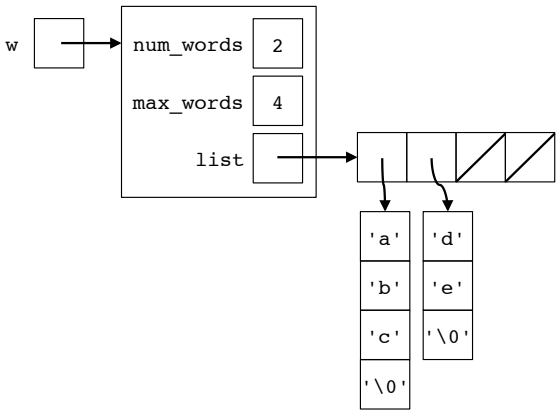
Write the implementation for a collection of words implemented as a dynamically allocated array of C strings. The collection can be populated using C strings containing words separated by spaces; there are various utility manipulation functions that can be performed on collections. The `Words` struct, function prototypes, and documentation about function definitions are given in `words.hpp`, which should not need to be changed. A stub implementation is provided in `words.cpp`; complete the implementation by filling in the areas commented `TODO`. You may change `words_main.cpp` as you like to try out your implementation.

Allocate and deallocate memory using the appropriate `new` and `delete` operators. You may include `<cstring>` and `<iostream>`. You may find the library functions `strlen`, `strcpy`, `strcmp`, and `strtok` useful but do not have to use them. Do not use `strdup` or C++ strings. For each C string in the container, allocate just enough memory to hold the string, and free that memory when the string is no longer needed in the container. Pointers that are not pointing to allocated memory should be set to `NULL`.

For example, after calling:

```
Words * w = newList(4);
appendList(w, "abc de");
```

The box and pointer diagram of `w` should look like:



Additionally, submit an image file that shows the box and pointer diagrams just for `w` after executing the following code:

a)
```
Words * w = newList("one two");
```

b)
```
Words * w = newList(2);
appendList(w, "removeme");
removeWord (w, "removeme");
```

c)
```
Words * w = newList("1 2");
Words * u = newList("3 4");
appendList(w, u);
```

You can use whatever method you like to create the image (Paint, Illustrator, sketch on paper and take a picture, etc.) but it should be submitted as a `.png` image named `diagrams.png`, with the appropriate letter next to each diagram.

Submission

Your submission should be a single zip file named `[LastnameFirstname]3.zip` including the following files:

```
Makefile
words.cpp
words.hpp
words_main.cpp
diagrams.png
```

For purposes of grading, assignments will be built and run on the CCIS Linux environment using `g++`. Assignments should include a makefile that builds all the program executables by default, and a `clean` target that removes everything but the source files and makefile. Assignments that are missing a makefile or a have makefile that does not build the programs will lose Style points.

Grading

Grading is broken down as:

- 40% - Functionality: Does the code handle inputs correctly? Does it handle error cases gracefully? Are corner cases accounted for? Does the program not crash?

- 30% - Implementation: Are the data structures and memory set up correctly? Is memory properly used and deallocated?
- 10% - Style: Is the code well-structured with appropriate functions? Are the variable names suitably descriptive? Does the code have explanatory comments? Is there a working makefile?
- 20% - Diagrams: Are the box and pointer diagrams correct?