CS 2123 Data Structures Recitation - Exercise 09

(Library – Improving bufferADT implementations with search/replace functions)

As in the previous recitations/hw, first you need to get some already implemented programs from the textbook. Specifically, get all the files under 09-Efficiency-and-ADTs from the class web page. You can get all by simply following the link "programs from the textbook" and then go to 00-zip-files and get 09-Efficiency-and-ADTs.zip and unzip it under directory X. I assume that you already got the booklib and installed it under directory X, as we described in previous recitations (if not, please follows the instructions at the end of this recitation to get booklib and 09-Efficiency-and-ADTs and save them under directory X)

Then go to 09-Efficiency-and-ADTs, and simply say make to compile editor.c program with different implementation of buffer ADT. Then run editor program.

After compiling/testing existing client/diver program called array-editor, stack-editor, and list-editor that uses different implementations of buffer ADT, you need to implement the following functions under **single linked list** in listbuf.c (you can try to implement them under other representations (i.e., array and stack) later at your own time).

void ReplaceCharInBuffer(bufferADT buffer, char oldch, char newch);

- When this function is called, it should start searching from the current **cursor** position, looking for the next occurrence of the character **oldch** in the *rest* of the buffer.
  - If it finds it, simply replace **oldch** with **newch**. Search should leave the **cursor** after the last character replaced in the buffer.
  - If **oldch** does not occur between the **cursor** and the end of the buffer, then there is nothing to replace and thus search should leave the **cursor** unchanged (in the original place)

int SearchStrBuffer(bufferADT buffer, char* str);

- When this function is called, it should start searching from the current **cursor** position, looking for the next occurrence of the string **str** in the rest of the buffer.
  - If it finds it, search should leave the **cursor** after the found string and return the value TRUE or 1
  - If the string **str** does not occur between the **cursor** and the end of the buffer, then search should leave the **cursor** unchanged and return FALSE or 0

- **First**, you need to add the above prototypes into buffer.h.
- **Second**, you need to implement them in listbuf.c
- **Third,** you need to update editor.c so that you can call those functions. So include the following in editor.c:
  - If user enters: **Rxy**, your editor.c should call **ReplaceCharInBuffer** (buffer, 'x', 'y') to replace all the occurrences of character 'x' with character 'y' after the cursor.
  - If user enters: **Sxyz**, your editor.c should call **SearchStrBuffer** (buffer, "xyz") to search string "xyz" in the buffer after the cursor.

To compile new version, use the following

```
> make list-editor
```

Note: other implementations array- and stack- will not work until you implement the above functions in them too. But their implementation is not part of this recitation. You can try to implement them later in your own time....

As always, make sure you release (free) the dynamically allocated memories if you allocate any memory in your programs. So, before submitting your program, run it with `valgrind` to see if there is any memory leakage…

Also if you need to debug your program, compile your programs with –g option and then run it with `gdb` and/or `ddd`.

What to return: !!!! NO LATE RECITATION ASSIGNMNET WILL BE ACCEPTED !!!

1. Create a directory called LASTNAME_Recitation09 and do all your work under that directory
2. First implement your program as described above...
3. Then compile and run it. Copy/paste the result in an output file, say out09.txt.
3. Finally zip your LASTNAME_Recitation09 directory as a single file LASNAME_Recitation09.zip and Go to BB Learn to submit it as **attachment** before the deadline.

/* Don't forget to include comments about the problem, yourself and each major step in your program! */

You must submit your work using Blackboard Learn and respect the following rules:

1) All assignments must be submitted as either a zip or tar archive file unless it is a single pdf file.
2) Assignments must include all source code.
3) Assignments must include an output.txt file which demonstrates the final test output run by the student.

If your assignment does not run/compile, the output.txt file should include an explanation of what was accomplished, what the error message was that prevented the student from finishing the assignment and what the student BELIEVES to be the underlying cause of the error.

Just in case you missed the previous recitations/hw, here is how to get the necessary files...

Go to class web page http://www.cs.utsa.edu/~korkmaz/teaching/cs2123/
Follow “programs from the textbook” link under online materials in the class webpage

get `booklib.zip` and `09-Efficiency-and-ADTs.zip`

save both under a common directory, say X in your local computer

Then follow these instructions:

```
~/X> unzip booklib.zip
~/X> cd booklib
~/X/booklib> make
~/X/booklib> cd ..

~/X>
~/X> unzip 09-Efficiency-and-ADTs.zip
~/X> cd 09-Efficiency-and-ADTs
~/X/09-Efficiency-and-ADTs> make
~/X/09-Efficiency-and-ADTs> make list-editor
```

Now you can run editor program.....