# 

|  |
| --- |
| **Academic Integrity** **You may NOT, under any circumstances, begin an assessment by looking for completed code on StackOverflow or Chegg or any such website, which you can claim as your own. Please check out the** [**Student Code of Conduct at PCC.**](https://www.pcc.edu/student-conduct/conduct/quick-view-of-policy/)  This assessment is individual work only. You must not discuss this assessment with any person other than your instructor. You may use course material for reference, but you must not use any other (non-course) material during this assessment.  I may ask you to explain your assessment verbally. If you cannot satisfactorily explain what your code does, and answer questions about why you wrote it in a particular way, then you should expect a zero. |

## 



This assessment focuses on the insertion of elements into an ordered linked list that has no duplicate entries.

This is a timed assessment, individual work only. Please do not discuss this assessment with any person other than the instructor.   
  
Please be sure to submit something before the deadline! Late submissions are not accepted, and will receive a zero score. Submitting something, even if it does not work as well as you would like, is always better than not submitting anything and receiving a zero score!

The files referenced in these instructions are available on the PCC Linux server in the directory:  
 /home/courses/doug.jones/162/as4\_p

You will need to copy these files to your development environment.

This assessment is similar in structure to Assessment 3, in that there are no classes, and the program creates a linked list of words entered by the user. A blank line (two consecutive ‘\n’ characters) marks the end of user input in this program (**not** EOF or Control-D as in Assessment 3).

The linked list contains no duplicates, and is ordered by string length in ascending order (short words first, long words last). If two words are the same length then they are ordered by ASCII value, which is the normal ordering for C++ std::string objects.

In this assessment you will implement two functions:

1. The print() function, which should display the nodes of the linked list in **reverse** order (last node in the list display first). For each node the print() function should disiplay the string value and the string length.
2. The remove() function, which should remove and delete all nodes in the list where the string length matches the parameter. In other words, if the parameter value is 4, the function would remove and delete all nodes in the linked list where the string has a length of 4.

You may **not** use any recursion in these functions.

A third function named update() is provided in this program, and you do not need to make any changes to the update() function. update() searches all nodes in the linked list for a specific substring, and then replaces that substring with a new value. You will see how update() works in the example below.

In the sample run shown below, the user enters words which are inserted into the linked list in ascending length order (short words first). The print() function displays the list in reverse order, so long words appear first in the display. After the list is displayed the program prompts the user for search and replacement strings, and then the update() function modifies the list (**without** reordering the list even if the string lengths change). Finally, the user is prompted for a string length and the remove() function removes all nodes that contain strings of that length.

> main

Enter input text (BLANK LINE to end):

hello world this is a test.

Hit the ENTER key twice to end the input

List of Words:

world (5)

twice (5)

test. (5)

input (5)

hello (5)

ENTER (5)

this (4)

the (3)

key (3)

end (3)

Hit (3)

to (2)

is (2)

a (1)

Enter search string: th

Enter replacement string: XX

Replacing "th" with "XX" in all list entries.

Updated List:

world (5)

twice (5)

test. (5)

input (5)

hello (5)

ENTER (5)

XXis (4)

XXe (3)

key (3)

end (3)

Hit (3)

to (2)

is (2)

a (1)

Enter target string length for remove: 5

Removing all entries with string length of 5.

Updated List:

XXis (4)

XXe (3)

key (3)

end (3)

Hit (3)

to (2)

is (2)

a (1)

>

## Tasks

* Copy the source code files (main.cpp main.h) from   
   /home/courses/doug.jones/162/as4\_p  
  to your development environment.   
    
  You do not need to develop your solution on the Linux server, but you must use these source code files from the Linux server as your starting point.
* This program creates a linked list of std::string objects. The node struct defined in main.h is the linked list node for a single link, forward linked list. Review the node struct definition in main.h, and note that the string in the node is dynamic, i.e. the node contains a pointer to the string.
* The main.cpp file creates an ordered linked list of words read from standard input (cin). The program reads input until the user enters a blank line (hit the ENTER key twice).
* The linked list is constructed so that
  + The list is ordered by string length, in ASCENDING order
  + The list contains no duplicate words
  + If two strings have the same length, then they are ordered by ASCII order
* If you are not familiar with C++ string objects you should review the zyBook information. The length of a string is given by the .length() method of the object, and the natural ordering of strings is ASCII, and the relational operators use this ordering for string data types.
* Edit the main.cpp file to implement the print() and remove() functions.. The parameter to the print() function is the list head pointer. The parameters to the remove() functions are the list head pointer and the string length to match for removal. s to this function are the linked list head pointer and a string to insert.
* You may develop your solution in any development environment, but you must test your solution on the PCC Linux Server
  + Copy all project files to a directory on the server
  + Use script to record the following steps in a file named output.txt:   
    script output.txt
    - Compile your project  
       g++ -o main main.cpp
    - Test your program on the PCC Server  
       ./main  
      valgrind --leak-check=full ./main
    - Exit the script  
      exit
  + Copy the output.txt and all source code files (.h and .cpp) from the server to your desktop
* Upload the output.txt and main.cpp files to the Desire2Learn dropbox for this practice assessment. **These are the only two files that will be graded**, so any additional files are just wasting space in D2L.

## Sample Run

> valgrind --leak-check=full main

==2426918== Memcheck, a memory error detector

==2426918== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.

==2426918== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info

==2426918== Command: main

==2426918==

Enter input text (BLANK LINE to end):

hello world this is a test

List of Words:

world (5)

hello (5)

this (4)

test (4)

is (2)

a (1)

Enter search string: es

Enter replacement string: XY

Replacing "es" with "XY" in all list entries.

Updated List:

world (5)

hello (5)

this (4)

tXYt (4)

is (2)

a (1)

Enter target string length for remove: 1

Removing all entries with string length of 1.

Updated List:

world (5)

hello (5)

this (4)

tXYt (4)

is (2)

==2426918==

==2426918== HEAP SUMMARY:

==2426918== in use at exit: 0 bytes in 0 blocks

==2426918== total heap usage: 18 allocs, 18 frees, 75,129 bytes allocated

==2426918==

==2426918== All heap blocks were freed -- no leaks are possible

==2426918==

==2426918== For lists of detected and suppressed errors, rerun with: -s

==2426918== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

>

TRADUCCION

Esta es una evaluación cronometrada, solo trabajo individual. No discuta esta evaluación con ninguna persona que no sea el instructor.

¡Asegúrese de enviar algo antes de la fecha límite! No se aceptan presentaciones tardías y recibirán una puntuación de cero. ¡Enviar algo, incluso si no funciona tan bien como le gustaría, siempre es mejor que no enviar nada y recibir una puntuación de cero!

Los archivos a los que se hace referencia en estas instrucciones están disponibles en el servidor PCC Linux en el directorio:

/inicio/cursos/doug.jones/162/as4\_p

Deberá copiar estos archivos en su entorno de desarrollo.

Esta evaluación tiene una estructura similar a la Evaluación 3, ya que no hay clases y el programa crea una lista vinculada de palabras ingresadas por el usuario. Una línea en blanco (dos caracteres '\n' consecutivos) marca el final de la entrada del usuario en este programa (no EOF o Control-D como en la Evaluación 3).

La lista enlazada no contiene duplicados y está ordenada por longitud de cadena en orden ascendente (las palabras cortas primero, las palabras largas al final). Si dos palabras tienen la misma longitud, se ordenan por valor ASCII, que es el orden normal para los objetos std::string de C++.

En esta evaluación implementará dos funciones:

1. La función print(), que debería mostrar los nodos de la lista enlazada en orden inverso (el último nodo de la lista se muestra primero). Para cada nodo, la función print() debe mostrar el valor de la cadena y la longitud de la cadena.

2. La función remove(), que debe eliminar y eliminar todos los nodos de la lista donde la longitud de la cadena coincide con el parámetro. En otras palabras, si el valor del parámetro es 4, la función eliminaría y borraría todos los nodos en la lista enlazada donde la cadena tiene una longitud de 4.

No puede utilizar ninguna recursividad en estas funciones.

En este programa se proporciona una tercera función denominada actualizar(), y no es necesario realizar ningún cambio en la función actualizar(). update() busca en todos los nodos de la lista vinculada una subcadena específica y luego reemplaza esa subcadena con un nuevo valor. Verá cómo funciona update() en el siguiente ejemplo.

En la ejecución de muestra que se muestra a continuación, el usuario ingresa palabras que se insertan en la lista vinculada en orden de longitud ascendente (primero las palabras cortas).

La función print() muestra la lista en orden inverso, por lo que las palabras largas aparecen primero en la pantalla. Una vez que se muestra la lista, el programa le pide al usuario que busque y reemplace las cadenas, y luego la función actualizar () modifica la lista (sin reordenar la lista, incluso si la longitud de la cadena cambia). Finalmente, se solicita al usuario una longitud de cadena y la función remove() elimina todos los nodos que contienen cadenas de esa longitud.

Tareas

● Copie los archivos de código fuente (main.cpp main.h) de

/inicio/cursos/doug.jones/162/as4\_p

a su entorno de desarrollo.

No necesita desarrollar su solución en el servidor Linux, pero debe usar estos archivos de código fuente del servidor Linux como punto de partida.

● Este programa crea una lista enlazada de objetos std::string. La estructura de nodo definida en main.h es el nodo de lista enlazada para una lista de enlaces directos de un solo enlace. Revise la definición de la estructura del nodo en main.h y tenga en cuenta que la cadena en el nodo es dinámica, es decir, el nodo contiene un puntero a la cadena.

● El archivo main.cpp crea una lista enlazada ordenada de palabras leídas desde la entrada estándar (cin). El programa lee la entrada hasta que el usuario ingresa una línea en blanco (presione la tecla ENTER dos veces).

● La lista vinculada está construida de modo que

● La lista está ordenada por longitud de cadena, en orden ASCENDENTE.

● La lista no contiene palabras duplicadas.

● Si dos cadenas tienen la misma longitud, se ordenan por orden ASCII.

● Si no está familiarizado con los objetos de cadena de C++, debe revisar la información de zyBook. La longitud de una cadena viene dada por el método .length() del objeto, y el orden natural de las cadenas es ASCII, y los operadores relacionales usan este orden para los tipos de datos de cadena.

● Edite el archivo main.cpp para implementar las funciones print() y remove(). El parámetro de la función print() es el puntero de encabezado de lista. Los parámetros de las funciones remove() son el puntero del encabezado de la lista y la longitud de la cadena para que coincida con la eliminación. Los elementos de esta función son el puntero de encabezado de la lista vinculada y una cadena para insertar.