COMP 322
Winter Semester 2022

INSTRUCTOR: DR. CHAD ZAMMAR
chad.zammar@mcgill.ca

Assignment 2: Exploring Dynamic Memory.

Due date: 11 Mars 2022, 11:59 PM.

Before you start:

- Research for similar problems on the internet is recommended. However, your submission should reflect individual work and personal effort.
- Some of the topics may not be covered in class due to our limited time. You are encouraged to find answers online. You can also reach out to the TAs for guidance.
- Please submit your assignment before the due date to avoid penalties or worse risking your assignment being rejected.
- Submit one file called **assignment2.cpp** containing all the functions together with their implementations. It will also contain the main() function that runs everything.

Make sure your code is clear and readable. **Readability of your code as well as the quality of your comments will be graded**.

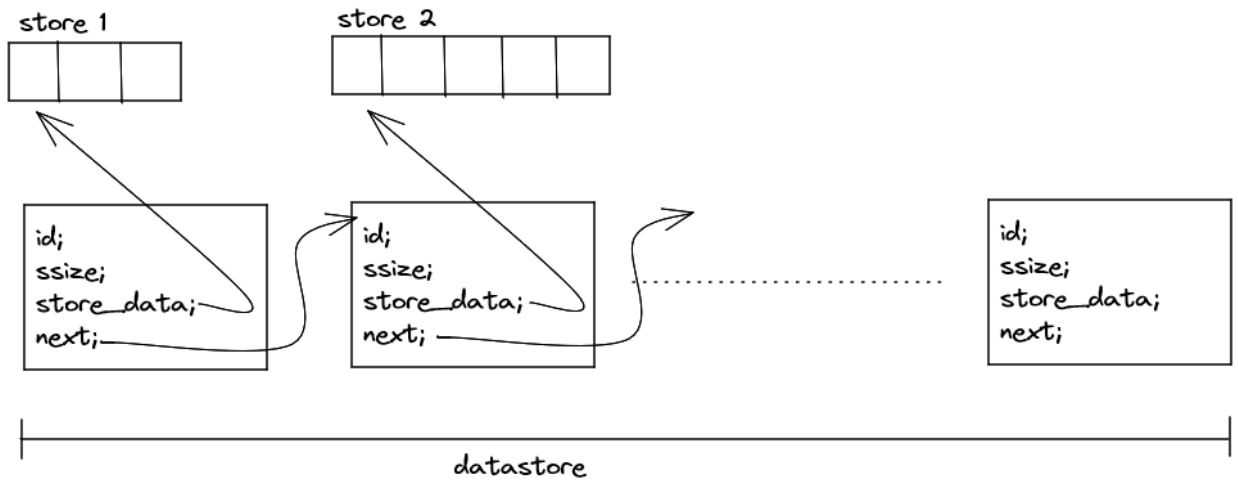
- No submission by email. Submit your work to mycourse.
- If your code does not compile it will not be graded.
- Be happy when working on your assignment, because a happy software developer has more inspiration than a sad one :).

In this assignment we will be enhancing the implementation of the first assignment by using dynamic memory management and pointers. This has the advantage of dynamically growing the size of datastore as the demand increases and also shrinking it dynamically when the demand decreases (when we delete a store).

This approach is much more efficient than the one used in the first assignment. There is no capacity cap like previously because we will be dynamically allocating as much memory as we need (until we hit the system's capacity in terms of RAM memory limit).

In this new implementation, each store is independent and may reside in totally different remote parts of memory. However, we still need to keep track of where the stores are, so we still need to maintain a linked list structure that we will be calling datastore. datastore will hold pointers to all the stores that were being dynamically allocated. Each cell of datastore is a structure holding multiple elements in it:

- The store's ID: *id*
- The store's size: *ssize*
- Pointer to the store's elements in memory: *store_data*
- Pointer to the next element in datastore: *next*



In other words, datastore is a linked list where each element is holding information about each store as well as the pointer to the store itself like described in the figure above.

Please reimplement all the questions from assignment 1. You should also be using the same main() function provided in assignment 1 for testing. The same output is expected.

In this implementation, we don't need to define extra arrays to hold store related data. This information is now being held by the structure maintaining information about each store.

The other difference from the first assignment is that we won't need to display the number of remaining available elements in the store because it is irrelevant since the store can extend dynamically to accommodate as many individual stores as needed. We can replace this display by the number of total used store elements at each time.

Please note that 10 points will be given for code readability and quality of comments provided.