Queue and Stack with singly-linked list nodes

You are given only a *main.cpp* file and an implementation of the data class *Measurement*. You are to implement the abstract data types (ADT) *Stack* and *Queue* so that the main() function compiles correctly. You must implement them from scratch and make sure you do that their declarations are in files called *"stack.h"* and *"queue.h"*, case sensitive (all lower case). You do not need to make any changes to *Measurement* and you do not need to instantiate instances of Measurement in your Queue or Stack, as they are instantiated in the main function and then sent (by value) into your data structure classes.
You must use singly-linked list nodes to implement your Queue and Stack functionality. You will have to implement a separate class for the list node itself, for a node with a single link (pointer) and an instance of *Measurement* as data.

In order for ***Stack*** to work correctly the following operations must be implemented:

- ***Stack()***
  A constructor with no parameters. It must set values for the class variables so that they represent an empty stack.
- ***~Stack()***
  The deconstructor. It must delete all memory that has been allocated by the class. That means all nodes in the list.
- ***void push(Measurement data)***
  The function adds one instance of measurement to the top of the stack.
- ***Measurement pop()***
  The function returns the value of Measurement from the top of the stack. It then removes the current top element.
  Throw an ***EmptyException*** *if* there is no data on the stack.
- ***int size()***
  Returns the number of elements currently on the stack.
- ***bool isEmpty()***
  Returns true if there are no elements on the stack, otherwise false.
- ***Overridden stream operator <<***
  Writes each data element to the stream with a single space (" ") between elements.

In order for ***Queue*** to work correctly the following operations must be implemented:

- ***Queue ()***
  A constructor with no parameters. It must set values for the class variables so that they represent an empty stack.
- ***~Queue ()***
  The deconstructor. It must delete all memory that has been allocated by the class. That means all nodes in the list.

- ***void add(Measurement data)***
  The function adds one instance of measurement to the back of the queue.
- ***Measurement remove()***
  The function returns the value of Measurement from the front of the queue. It then removes the current front element.
  Throw an ***EmptyException*** *if* there is no data in the queue*.*
- ***int size()***
  Returns the number of elements currently in the queue.
- ***bool isEmpty()***
  Returns true if there are no elements in the queue, otherwise false.
- ***Overridden stream operator <<***
  Writes each data element to the stream with a single space (" ") between elements.

You must declare the class ***EmptyException*** yourselves. You can see how this is done in previous assignments. I would declare it in the .h file for your singly-linked node, as that file will likely be included in both classes. If this doesn't fit your implementation, that is OK, as long as the exception is declared and used correctly.

You can choose whether you include all node connecting functionality within each class (Queue and Stack) or if you implement a separate class, *SinglyLinkedList*, which offers functionality to add to each end of the list and remove from one end, and then use an instance of that class within each of the other classes. So long as singly-linked nodes are used.

The file **ExpectedOutput.txt** shows the correct output from the current main function.

The assignment will be returned through Mooshak.

- You get ***three attempts*** to enter it into Mooshak. The third one is final.
- Send a ZIP folder into Mooshak containing your code, .cpp and .h files.

You must make really good tests in your own main function.

- The current main function contains helper functions and examples of their use.
- You must make sure that you test all edge cases.
  - Try adding and removing in different orders
- Be absolutely sure that your code works and that you understand why it works before trying to understand Mooshak's error messages.
- Make sure your program doesn't leak memory.
  - For every new there should be a delete.
  - Deconstructor should delete everything that was allocated.

Röð og stafli með eintengdum hnútum

Einungis er gefin main skrá og útfærsla á gagnaklasanum *Measurement*.
Verkefnið er að útfæra hugræna gagnatögin (ADT) *Stack* og *Queue* á þann hátt að main() fallið virki rétt. Það verður að útfæra klasana frá grunni og ganga úr skugga um að yfirlýsingar séu í skrám sem heita *“stack.h”* og *“queue.h”*, með litlum stöfum.
Það þarf ekki að gera neinar breytingar á *Measurement* og það þarf ekki að búa til tilvik af Measurement inni í Queue eða Stack klösunum þar sem tilvikin eru búin til í main fallinu og gildi þeirra send inn í gagnagrindurnar ykkar.
Það verður að nota eintengda listahnúta til að útfæra virkna raðarinnar og staflans.
Það þarf að útfæra sérklasa fyrir hnútinn sjálfan, sem inniheldur eina tengingu (bendi) og tilvik af Measurement sem eru gögnin sem hnúturinn geymir.

Til þess að ***Stack*** virki rétt þurfa eftirfarandi aðgerðir að vera útfræðar:

- ***Stack()***
  Færibreytulaus smiður. Hér verður að setja gildi á klasabreytur þannig að þær lýsi tómum stafla.
- ***~Stack()***
  Eyðirinn. Hér verður að eyða öllu minni sem klasinn hefur úthlutað, þ.e. öllum hnútunum í listanum.
- ***void push(Measurement data)***
  Fallið bætir einu tilviki af Measurement efst á staflann.
- ***Measurement pop()***
  Fallið skilar gildinu af Measurement tilvikinu sem er efst á staflanum. Síðan fjarlægir það stakið af staflanum.
  Kastið ***EmptyException*** ef það eru engin gögn á staflanum.
- ***int size()***
  Skilar fjölda staka sem eru á staflanum.
- ***bool isEmpty()***
  Skilar true ef engin stök eru á staflanum, annars true.
- ***Overridden stream operator <<***
  Skrifar hvert stak úr listanum á strauminn, með einfalt bil (“ “) á milli staka.

Til þess að ***Queue*** virki rétt þurfa eftirfarandi aðgerðir að vera útfræðar:

- ***Queue ()***
  Færibreytulaus smiður. Hér verður að setja gildi á klasabreytur þannig að þær lýsi tómri röð.
- ***~Queue ()***
  Eyðirinn. Hér verður að eyða öllu minni sem klasinn hefur úthlutað, þ.e. öllum hnútunum í listanum.

- ***void add(Measurement data)***
  Fallið bætir einu tilviki af Measurement aftast í röðina.
- ***Measurement remove()***
  Fallið skilar gildinu af Measurement tilvikinu sem er fremst í röðinni. Síðan fjarlægir það stakið úr röðinni.
  Kastið ***EmptyException*** ef það eru engin gögn í röðinni.
- ***int size()***
  Skilar fjölda staka sem eru í röðinni.
- ***bool isEmpty()***
  Skilar true ef engin stök eru í röðinni, annars true.
- ***Overridden stream operator <<***
  Skrifar hvert stak úr listanum á strauminn, með einfalt bil (“ “) á milli staka.

Það þarf að lýsa yfir klasanum ***EmptyException***. Það má sjá hvernig þetta er gert í fyrri verkefnum. Ég myndi lýsa þessu yfir í .h skránni fryir eintaengda hnútinn, þar sem sú skrá verður að öllum líkindum aðgengileg (með #include) í báðum klösunum. Ef þetta passar ekki við ykkar útfærslu þá er það allt í lagi, svo fremi frávikinu sé lýst yfir og það rétt notað.

Það má velja hvort virknin sem tengir saman hnúta er útfærð í hvorum klasanum (Queue og Stack) fyrir sig eða hvort útfærður er sérklasi, *SinglyLinkedList*, sem býður upp á virkni til þess að bæta staki á hvorn enda listans og taka það af öðrum endanum, og tilvik af honum síðan notað í hinum klösunum tveimur til að sjá um virknina. Aðalmálið er að eintengdir hnútar séu notaðir.

Skráin **ExpectedOutput.txt** sýnir rétt úttak úr núverandi main() forriti.

Verkefninu er skilað á Mooshak.

- Þið fáið ***þrjár tilraunir*** til að senda verkefnið inn í Mooshak. Þriðja tilraunin er lokaskil.
- Sendið ZIP skrá inn í Mooshak sem inniheldur allan forritstextann ykkar, .cpp og .h skrár.

Þið verðið að gera virkilega góðar prófanir í main fallinu.

- Núverandi main() forrit inniheldur hjálparföll og dæmi um notkun þeirra.
- Þið verðið að sjá til þess að jaðartilvik séu prófuð.
  - Bætið í gagnagrindina og takið úr henni í mismunandi röð.
- Verið alveg viss um að ykkar forritstexti virki og að þið skiljið hvernig og hvers vegna hann virkar áður en þið byrjið að reyna að skilja villuskilaboðin frá Mooshak.
- Verið viss um að forritið leki ekki minni.
  - Fyrir hvert new ætti að vera delete.
  - Eyðirinn ætti að eyða öllu minni sem hefur við úthlutað.