COP4530 – Project 1

Most Uniform SubSequence

Submission:

1. Create a tar file with the required five files (see Section 3).
2. Make an electronic submission of your tarball file on Canvas.

Due Date: September 24, 2019, 11:50pm for all students
Late Penalty: maximum one day at the penalty of 15%.

1. Overview

The objectives of this project include the following:

- Get refreshed with C++ features;
- Get familiar with using function and class templates; and
- Demonstrate your understanding of recursion and algorithmic complexity.

2. Basic Description

For this assignment you will implement a templated class `NSequence` (New Sequence). It is similar to `Vector` but with a different set of member functions. NSequence can take a primitive data type as its item, such as `int, string`, or other class types, such as `IntCell` as discussed in class. The following starter files are provided for you. You are not supposed to include them in your final submission. Thus any change you make to these files will not be graded.

- `Cell.h` -- fully defines an item class (IntCell) for use in a NSequence. It is very similar to the IntCell class we discussed in class. You need to provide the implementation of all member functions, except <<, in a new file called `Cell.cpp`.
- `NSequence.h` -- provides initial declarations of the template class NSequence. You need to implement all member functions. You can view them as separated groups: (1) Big Five and the explicit constructor; (2) the accessors (isEmpty, getSize, getCapacity, [], getFirst, getLast); (3) the mutators (insert, remove, push, pop, and etc); and (4) the utility functions (printout and growCapacity). Your implementation should be named as `NSequence.hpp`.
- `FunClassTemp.h` -- provides initial declarations of the template functions and template classes. You need to implement all the functions in a separated file (`FunClassTemp.hpp`). You can view them two different groups: (1) three functions for `findDiff` and `getDiff`; and (2) four functions that search in a NSquence for the longest subsequence whose items fall within the range of `[ref-range, ref+range]`. Note that the non-primitive datatype IntCell does not have an add operator. These four functions should have the complexity of O(N3), O(N2), O(N) and O(NlogN), respectively.
- `Drivers` – three separate driver programs are provided for you to perform unit tests on your code: TestCell.cpp, TestSeq.cpp and MaxSubRange.cpp.
    - `TestCell.cpp` tests your implementation of member functions for the class IntCell, and the group of functions (findDiff and getDiff) declared in FunClassTemp.h.
    - `TestSeq.cpp` tests your implementation of member functions for the template class NSequence, and the function of the class IntCell when it is to instantiate NSequence.
    - `MaxSubRange.cpp` tests your implementation of search functions in a NSequence for the longest subsequence as defined before with different complexities.

- `sample.output` -- contains the output from running a Makefile and the three driver programs as mentioned above.
- `README` – a short reference on the main files you should submit and the grading rubrics.

3. Programs and files you need to create.

You need to provide a total of five files for the completion of this assignment.

- `Cell.cpp`: as described in Section 2
- `NSequence.hpp`: as described in Section 2
- `FunClassTemp.hpp`: as described in Section 2
- `<fsuid>_driver.cpp`:
  - MaxSubRange.cpp currently only tests the search functions for a NSquence with integers as member items. You need to change the driver MaxSubRange.cpp to create a NSequence object with IntCell as member items, and then test the four search functions, and print out the item values for validation. Your file should be named as <`fsuid`>_driver.cpp, where <`fsuid`> denotes your FSU email ID. For example, my file would be wyu3_driver.cpp if I were to complete this assignment.
- `Makefile`:
  - Create a Makefile that builds all the provided test programs and your own test program (<fsuid>_driver.cpp), i.e., when you type "make" in the directory, it should compile and build four executables (TestCell, TestSeq, MaxSubRange, and <fsuid>_driver).

On linprog machines, you can create a tarball file from these five files using the following command (all characters in one line), where `fsuid` is your FSU ID.

- `tar zcf <fsuid>_proj1.tgz Cell.cpp NSequence.hpp FunClassTemp.hpp <fsuid>_driver.cpp Makefile`

4. Details on IntCell and the template functions and classes

Here are general descriptions of the two classes you are to define, along with a general description of each function's task.

A. class IntCell

Member variable descriptions

- `storedValue:` an integer stores the value of the object. Since it is not a pinter, it has no complication of deep copy versus shallow copy.

Function descriptions

- `Big Five`
  - No change on the default big five.
- `explicit constructor`
  - explicit Constructor – create a new object with an initialValue, use 0 by default if no input provided.
- `Accessors`

- size/read – both return the value of storedValue.

- Mutators
  - write – update the value of the member variable - storedValue.
- Operators
  - Operator –: subtract the value of another IntCell object, and returns the difference.
  - Operator <: compare the value of two IntCell objects.
  - Friend operator <<: output the value of IntCell to the osteam out.

B. Template class NSequence

This template class has three member variables: numOfItems, totalCapacity, and a pointer to an array of items. This template class can be instantiated with int, string, or IntCell.

Member variable descriptions

- `items:` a pointer to an array of objects of typename T.
- `totalCapacity:` an integer records the total number of objects pointed by items. Not all objects have been initialized.
- `numOfItems:` an integer keeps track of the total number of available items. Any constructor will create at least one item in a new NSequence object.

Function descriptions

- Big Five and explicit constructor
  - explicit Constructor – create a new NSequence with at least initSize items. If initSize is non-positive, one is automatically used.
  - Destructor -- appropriate clean-up of list, no memory leaks
  - Copy constructor -- deep copy
  - Copy assignment operator -- deep copy
  - Move constructor -- Constructor with standard move semantics
  - Move assignment operator -- assignment with standard move semantics
- Accessors
  - isEmpty -- returns true if the NSequence is empty, false otherwise
  - getSize -- returns the size (number of data elements) in the items.
  - getCapacity - returns the capacity of the NSequence.
  - getFirst -- returns the first element in the sequence (by reference)
  - getLast -- returns the last element in the sequence (by reference)
  - const operator [] -- returns the element at the position (index) by reference.
  - This operator is overloaded to return either a mutable item or a const item.
- Mutators
  - insert – insert an item before the item as position (pos). It is overloaded to allow both copy and move semantics.
  - remove -- remove an item before the item as position (pos).
  - push_back - insert an item at the end position. It is overloaded to allow both copy and

move semantics.

- pop_back - remove the item at the end position.
- operator [] -- returns the element at the position (index) by reference.

- Utility functions
  - growCapacity – grow the capacity of NSequence by the specified parameter. If zero input is provided, the capacity is doubled.
  - printOut – print out the value of up to 50 items in the range of [begin, end].

5. General Requirements

- Document your code appropriately so that it is readable and easy to navigate
- You may use standard C++ I/O libraries, as well as class libraries like string. You may NOT use any of the container class libraries from the STL for the implementation of NSequence and IntCell.
- Make sure your files compile and run on linprog.cs.fsu.edu with g++, using the C++11 standard compilation flag. Your Makefile should function in a similar manner to produce the driver programs, as shown in the file sample.output. Your code will be tested and graded on linprog machines.

6. Breakdown of points

The grade breakdown for the project is as follows:

Points	Requirements
5	Have completed the required functions in Cell.cpp, NSequence.hpp and FunClassTemp.hpp in a reasonable correct manner.
5	The submitted files Cell.cpp, NSequence.hpp and FunClassTemp.hpp can compile via a Makefile and produce driver programs for the provided three tests.
15	Correct implementation of Cell.cpp (5 points); correct implementation of getDiff and findDiff (5 points), and successful passing of TestCell (5 points).
30	Correct implementation of NSequence.hpp, including 10 points for the Big Five and explicit constructor, 10 points for the mutators, 5 for the accessors, and 5 for correct passing of TestSeq
30	Implementation of search algorithms. 5 each for cubic and quadratic solutions, 10 each for O(N) and O(NlogN) solutions
10	Correct implementation of a driver program to test NSequence instantiated with IntCell. Your sample output should contain a subsequence with at least 3 items.
5	Your new driver produces output in a format similar to the output of MaxSubRange. This requires you to use the printout function.

7. Miscellaneous

The first person to report any compilation or execution errors in the provided materials will be given 3% extra credit (maximum 15% per person). We have explicitly separated your code from the provided source code files. Thus Automatic plagiarism detection software will be used on all submissions – any cases detected will result in a grade of 0 for those involved.