HBM803E - Object Oriented Programming Techniques

HOMEWORK IV (due to Saturday 23/01/2021 23:59 PM)

Questions

1. (**20 pts**) Start with a list of int values. Use two normal (not reverse) iterators, one moving forward through the list and one moving backward, in a while loop, to reverse the contents of the list. You can use the swap() algorithm to save a few statements. (Make sure your solution works for both even and odd numbers of items.)

2. **(20 pts)** Fill an array with even numbers and a set with odd numbers. Use the merge() algorithm to merge these containers into a vector. Display the vector contents to show that all went well.

3. **(16 pts)** This question is quite similar to fourth question of homework 3, but it is less detailed and you will use STL containers instead of arrays (array: pointer and dynamic memory allocation). Write a class **Matrix_COO** which has:
   - *2 vector<int> and 1 vector<double> attributes to store the matrix.* One of integer vectors will hold row index of i-th element at its i-th position, the other integer vector will hold column index the same way. The vector that holds double precision floating point number data will be used to store values of non-zero matrix elements.
   - 2 integer attributes to store number of rows and columns of the whole matrix.
   - A constructor that takes a file name (actually path) as argument. That constructor should open the file at the path in reading mode, read and store the data **without counting NNZ** (number of non-zero elements) in the first place as we do not need that information before memory allocation unlike hw-3 version. Input file format will be same as hw-3, and q3-input.txt which is already provided with hw-3 can be used for test.

4. **(16 pts)** The same question as question 3 of this homework, but uses map-based data structure explained below instead of 3 vectors used in the previous question:
   - A map attribute with integer key and map values. The map, which is used as value element in the map attribute, has an integer key and double value. Use integer key of the bigger map (which is used directly as attribute of the class) for row indices of non-zero elements. As you can conclude from that, there will be *only one key-value pair* in this map *for each nonempty row* in the matrix. The integer key value of the smaller map (which is used as value in every key-value pair of the bigger map) will be used to store column indices of non-zero elements at that row. At the end, it can be

called a 2 dimensional map.

- A similar constructor to question 3, but that stores elements in the aforementioned map based structure.

Note that exception handling parts or any other **parts of the hw-3 version** that is not mentioned here **are not required.** But you can use your own answer to hw-3 q-4 and modify it to use with STL containers for questions 3 and 4, but only aforementioned parts will be considered in the evaluation. If you did not already write a working code for related questions in hw-3, you do not need to include these parts.

5. **(3 pts)** Consider the differences of 3 COO class implementation (q-4 in hw-3, q-3 and q-4 in hw-4). All of them does the same job, but uses different data structures. Search for and think about possible advantages and disadvantages of this 3 different data structure we used to store a sparce matrix, especially for handling matrix-vector and matrix-matrix operations whose performances are usually limited with cache misses when reaching to / searching for the matrix elements.
Compare their expected performances (sort with = and >) of this 3 implementation based on your understanding after your research, and explain your answers with one or two sentences:
- Random access to matrix elements (with specified indices)
- Access to all elements in a row
- Access to all elements in a column
- Matrix-vector multiplication (dot product)
- Matrix-matrix product

6. **(20 pts)** Implement the following methods to the Matrix_COO class you chose (you can skip the reading and constructor part if you did not answer any related question, and construct one simple example matrix with ={3, 5, 6,…} notation for initialization)
- + operation, that add two Matrix_COO objects of the same implementations (you may need copy constructor for returning answer) **(5 pts)**
- .dot( ) method function that handles dot product of two Matrix_COO instances **(10 pts)**
- operator overloading for * operator that takes an instance of Matrix_COO class as argument and only calls .dot( ) method and returns its return value. **(1 pts)**
- operator overloading for * operator that takes a double as argument and multiplies every element of "(*this)" (which is the object that * is called in, or the object at its left in main function) with that double type argument. **(2 pts)**

- friend operator * for handling operations with a double type variable at the left of * and Matrix_COO instance at the right of * operator when it is called. **(2 pts)**

7. **(5 pts)** Search for the following and summarize with a few sentences:
   - the keyword *auto* and function *decltype* **(2 pts)**
   - header "bits/stdc++" and what disadvantage it has **(2 pts)**
   - Difference between the square brackets operator [ ] and .at( ) method of vector container in the STL. **(1 pts)**

Rules

- As with all programs in this course, your code should contain useful comments. In particular, your name, the date, and a brief description of what the program does should appear at the top of your source file.

- Each question should have its own folder (named with q1, q2, q3 etc.), which contains source code and Makefile to compile it. The homework can be given as an Eclipse project but in this case each question will be defined as different project.

- All files must be packed and compressed with following naming convention,

  hw-04-[STUDENT-ID].tar.gz

  The created file must be uploaded to http://ninova.itu.edu.tr/tr/