Enel 452 – Fall 2024 – Assign 2 – Linked List

Handed Out: [2024-10-01 Tue]
Due: [2024-10-08 Tue 23:55]

Using the C++ linked-list queue implementation (see attached source files) as a starting point, modify it to allow an element to be inserted at any point in the queue, rather than just at the tail. The insertion point should be specified with an `unsigned` parameter, with value from `0` to `size()`, with the new element inserted just *before* the specified position.

Requirements

- You must implement this method:

  ```
  void insert(Data d, unsigned position);
  ```

  The position is a number from 0 to `size()`, and the data element `d` will be inserted *in front of* that queue position. That is, if position is 0, it will insert at the front of the queue (the head), if you set position to `size()-1` it will insert just in front of the tail, if you set position to `size()` the element will be appended at the tail (just as a normal `insert` function does).

- I'll expect your code to simply link with mine with no problems, and to pass my test drive code. You must conform to the specified interface. Below is an example of test code that should work.

  ```
  Queue q;
  q.insert(Data(1,2));
  q.insert(Data(3,4));
  q.insert(Data(5,6));
  q.insert(Data(-2,-3), 1);
  q.insert(Data(-4,-5), 3);
  q.print();
  // should produce 0:(1,2) 1:(-2,-3) 2:(3,4) 3:(-4,-5) 4:(5,6)
  ```

- if an illegal insertion is attempted, the program should print (to the stderr)

  ```
      insert: range error.
  ```

  and terminate with an exit code of 3.

- Write a test function that tests the operation of your `insert`.

- You may add other unit tests if you like.

- Your code must compile and run on a target like `snoopy`. You can develop and test it on any platform you like, but I'll test it on my own machine (which also runs Linux).

  - Exception: if you develop for the nucleo-64 f103rb platform, then I'll test it on that platform.

- Write up a short readme document

  - describe the addition you made (which should be clearly marked in the source code)
  - describe any unit tests you added
  - anything else that should be mentioned.

- This code should compile easily under g++ or ANY C++ compiler. There's no exotic stuff we're doing here.

- Submit your readme, implementation, and interface files, to your git repo.