Assignment 5 Part 2 Grade Report (Page 1 of 1)

Name: *Shackelford_Jade* **Due:** 03/11/2022 (11:55 pm)
Received: on-time ~~03/ /2022 (:)~~

Remarks: compilation: compiled OK / ~~compiled with warning(s)~~ / wouldn't compile
test w/ a5p2_test.in: ~~OK / crashed out~~ / (N/A)
code check: (see ***penalty tally*** below)

Score: ~~(50 − (from below)) / 50 =~~ 15 / 50 (partial credit)

■ **Not Heeding Style Guide (up to 30% of total points)**

- *Naming Convention:*
  - not using meaningful name
  - not starting a variable name in lowercase
  - not using separating underscore(s) or camel style in multi-word name
  - name of constant not in all-uppercase or name of variable in all-uppercase
  - function not doing, or doing more than, what it's name suggests
- Other:
- *Readability Woes:* (penalty - depends on severity)
  - poor indentation
  - poor spacing
  - poor alignment
  - lines wrap/cut off
  - using 1 or o (looks like 1 or 0) as variable name
  - not using monospaced font

■ **Not Doing According to Requirement/Specification and Other General Shortcomings**

- Not implementing `ShowAll_BF` using `cnPtrQueue` as required. [10]
- Dangling pointer(s). [½]
- Other:

■ **Function Specific Issues: `cnPtrQueue()` (default constructor)**

- Not using initializer list (where possible). [½]
- Not initializing member variables (but declaring/initializing 3 local variables instead). [2]
- Not initializing `numItems`. [1]
- Not implementing in `cnPtrQueue.cpp` but including it in `cnPtrQueue.h` instead). [1]
- Creating 2 unused local stacks. [1½]
- Not implementing at all [2½]
- Other:

■ **Function Specific Issues: `size()`**

- Not implementing at all [3]
- Using `size_t` or defining another `size_type` instead of `cnPtrQueue::size_type` [½]
- Declaring separate `size_type` in implementation that can conflict `cnPtrQueue::size_type` [½]
- Not accounting for items in 1 of the stacks. [1]
- Other:

■ **Function Specific Issues: `empty()`**

- Other:

■ **Function Specific Issues: `front()`**

- Not or not correctly checking precondition. [½ 1 1½ 2]
- Other:

■ **Function Specific Issues: `push()`**

- Not incrementing `numItems`. [1]
- Not using `inStack` and `outStack` at all (do insert at head to a linked list instead)
- Other:

■ **Function Specific Issues: `pop()`**

- Not or not correctly checking precondition. [½ 1 1½ 2]
- Not correctly checking precondition: ANDing instead of ORing the 2 conditions. [1½]
- Not decrementing `numItems`. [1]
- Other:

■ **Function Specific Issues: `ShowAll_BF()`**

- Not implementing algorithm correctly (closing parenthesis for 1st while misplaced) - incorrect output. [5]
- Doing `pListHead->data` (to give initial value) w/ `pListHead` possibly null - will crash if `pListHead` is 0 (empty list). [1]
- Initializing local pointer (`cursor`) with `new CNode` instead of 0 - memory leak. [½]
- Using a dynamically allocated cnPtrQueue but not freeing up dynamic memory at the end - memory leak. [1½]
- Using `cout` instead of `outs`. [½]
- Other:

■ **Others:**

- Adding/implementing not quite meaningful destructor. [½]
- Preceding `cnPtrQueue::size_type` with `typename` -> GCC compilation error (CS Linux). [1]

```
#include "nodes_LLoLL.h"
#include "cnPtrQueue.h"
#include <iostream>
using namespace std;

namespace CS3358_SP2022_A5P2
{

  // do breadth-first (level) traversal and print data
  void ShowAll_BF(PNode* pListHead, ostream& outs) {
    cnPtrQueue queue;
    while (pListHead != 0) {
      CNode* cListHead = pListHead->data;
      while (cListHead != 0) {
        queue.push(cListHead);
        cListHead = cListHead->link;
      }
      pListHead = pListHead->link;
    }
    while (!queue.empty()) {
      outs << queue.front->data << " ";
      queue.pop;
    }
  }
```

X

```
#include "cnPtrQueue.h"
#include <cassert>
using namespace std;

namespace CS3358_SP2022_A5P2 {

  cnPtrQueue::cnPtrQueue() {

  }

    bool cnPtrQueue::empty() {
      return (outStack.empty() && inStack.empty());
    }

    size_type cnPtrQueue::size() return numItems;

    CNode* cnPtrQueue::front() {
      assert( !inStack.empty());
      if ( outStack.empty() ) {
        while ( !inStack.empty() ) {
          outStack.push( inStack.top() );
```

not init. numItems

compilation error

X

```
            inStack.pop();
        }
      }
      return outStack.top()
    }

    void cnPtrQueue::push(CNode* cnPtr) {
      inStack.push(cnPtr);
    }

    void cnPtrQueue::pop() {
    assert( !inStack.empty());
      if ( outStack.empty() ) {
        while ( !inStack.empty() ) {
          outStack.push( inStack.top() );
          inStack.pop();
        }
      }
      outStack.pop();
    }
}
```

not updating numItems