Lab 09 Queues

Exercise 1: Creating and using queues with link-lists

Complete the program code below to simulate students queuing at a faculty office. There is only one queue for faculty staff dealing with the students. Print out the students in the queue every time a student joins the queue or leaves the queue after dealing with the faculty staff. Use `Queue` ADT class to simulate the queue.

The following code for the `Student` class is provided for you to use. The node class definitions and declarations are provided as .h/cpp files and are the same ones from Lab 7.

```
class Student {
  string name;
  int id;
 public:
  Student (string name = "", int id = 0) : name(name), id(id) {}
  friend ostream& operator<< (ostream& os, Student& s) {
       os << s.name << "/" << s.id;
    return os;
  }
};
```

Start by creating the ADT queue class and completing the methods as listed below

```
template<class ItemType>
class LinkedQueue{
private:
   Node<ItemType>* backPtr;
   Node<ItemType>* frontPtr;
public:
   LinkedQueue();
   ~LinkedQueue();

       bool isEmpty() const;
       bool enqueue(const ItemType& newEntry);
       bool dequeue();

       ItemType peekFront() const;

       template<class ItemType>
       friend ostream& operator<< (ostream& o,LinkedQueue<ItemType>& i);
};
```

To test the queue class, the following main driver program is provided for you to complete

```
int main(){
  LinkedQueue<Student> queue;
  Student s;
  string name;
  bool cont=true;
  int id;
  int choice;
  do {
    cout << "Queue: " << queue << endl;
       cout << "Option:\n"
            << "1: Enqueue a student\n"
            << "2: Dequeue a student\n"
            << "3: Peek\n"
```

```
            << "4: Quit\n";
       cin >> choice;
     switch (choice) {
       case 1: // ask for student info and add to queue
       case 2: // dequeue the student
       case 3: // show student at front of queue
       case 4: // show student at front of queue
       }
  } while (cont);
}
```

Exercise 2 : Creating queues using arrays

Repeat exercise 1 but make use of arrays to replace the ADT queue class.

Note: The queue uses link lists which avoid the problems of having to "move" items up the queue to make room for new items to enter the list. How would you change the code to make use of arrays instead?

Extra Exercises

Question 1: Animal Card Game

Animal Card Game is a simple 2-player card game that consists of 6 cards (2 Mouse cards, 2 Snake cards, and 2 Elephant cards). Each animal is superior to another animal based on the following rule:

1) **Elephant** is superior to **Snake**
2) **Snake** is superior to **Mouse**
3) **Mouse** is superior to **Elephant**

Initially, each player holds three cards. These three cards may not be unique (i.e. repeated animals is possible). At each round of the game, both players display their first card. The player whose first card is more superior captures his opponent's first card, which is done by keeping his opponent's first card at the bottom of his pile of cards (lose card), followed by his first card (win card).

If both of their first cards are the same (i.e. they draw), their first cards will be put in the standby location. Then, both players will keep displaying their next first card until one player captures the other. When that happens, all the cards in the standby location will belong to the winning player. Then, his opponent's first card will again be put at the bottom of his pile of cards, followed by his first card. The game ends when one of the player losses all his cards, or when both players has no cards left (i.e. it's a tie).

Some examples of the game are provided as follows:

Example 1:

(M = Mouse, S = Snake, E = Elephant)
Initial Setup:
Player 1: S, M, E
Player 2: E, M, S

Standby:

1st Round:
Player 2 Elephant captures Player 1 Snake
After capture,
Player 1: M, E
Player 2: M, S, S, E
Standby:

2nd Round:
Player 1 and Player 2 draws.
Both Mouse cards are put in standby location.
After draw:
Player 1: E
Player 2: S, S, E
Standby: M, M

3rd Round:
Player 1 Elephant captures Player 2 Snake.
All cards in standby location belong to Player 1.
After capture,
Player 1: M, M, S, E
Player 2: S, E
Standby:

4th Round:
Player 2 Snake captures Player 1 Mouse
After capture,
Player 1: M, S, E
Player 2: E, M, S
Standby:

5th Round:
Player 1 Mouse captures Player 2 Elephant
After capture,
Player 1: S, E, E, M
Player 2: M, S
Standby:

6th Round:
Player 1 Snake captures Player 2 Mouse
After capture,
Player 1: E, E, M, M, S
Player 2: S
Standby:

7th Round:
Player 1 Elephant captures Player 2 Snake
After capture,
Player 1: E, M, M, S, S, E
Player 2:
Standby:

Player 1 wins.

Example 2:

(M = Mouse, S = Snake, E = Elephant)
Initial Setup:
Player 1: E, M, S
Player 2: E, M, S
Standby:

1st Round:
Player 1 and Player 2 draws.
Both of the Elephant cards are put in standby location.
After draw,
Player 1: M, S
Player 2: M, S
Standby: E, E

2nd Round:
Player 1 and Player 2 draws.
Both of the Mouse cards are put in standby location.
After draw,
Player 1: S
Player 2: S
Standby: E, E, M, M

3rd Round:
Player 1 and Player 2 draws.
Both of the Snake cards are put in standby location.
After draw,
Player 1:
Player 2:
Standby: E, E, M, M, S, S

Tie.

Tasks:
Write a program that accepts the cards for both players as shown in the sample runs below. Based on the input, the program should show "**Player 1 wins.**", "**Player 2 wins.**", or "**Tie.**". Remember to remove [App_q1.cpp] from the project and create a new empty file named [App_q2.cpp] into the project to start your Animal Card Game implementation.

Sample Run 1:

```
Enter Player 1's cards: S M E
Enter Player 2's cards: E M S
Player 1 wins.
```

Sample Run 2:

```
Enter Player 1's cards: E M S
Enter Player 2's cards: E M S
Tie.
```

Sample Run 3:

```
Enter Player 1's cards: E E M
```

```
Enter Player 2's cards: S M S
Player 2 wins.
```

Question for you to ponder:

Will the Animal Card Game run forever? Explain your answer.

Note that, the user input as shown below is not allowed because the game only consists of 2 Mouse, 2 Snake, and 2 Elephant cards.

Enter Player 1's cards: E M S
Enter Player 2's cards: S S E