CMSC 341 Data Structures
Stack, Deque and Queue Review

These are some review questions on stacks, deques and queues. Note that deques are NOT covered in the textbook. The class definitions for stack, deque and queue are provide at the end of the questions.

Stacks

1. Using only the operations of the stack, write a function that determines if a string is a palindrome (i.e. reads the same backward and forward; e.g. "level"). The prototype for this function is given below.

```
bool IsPalindrome( const string& theString );
```

2. What is the output of the following code?

```
int values[10] = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19 };
Stack< int > s;

for (int 1 = 0; i < 10; i++)
     s.push( values[ i ] );

int n = 25;
for (int i = 0; i < 4; i++)

{
     n += s.top( ); s.pop( );
}
for (int i = 0; i < 2; i++)
{
     n -= s.top( ); s.pop( );
}
cout << n << endl;
```

1. Discuss the advantages and disadvantages of the text's array implementation and the lecture notes layered implementation of the stack ADT. At a minimum, consider the asymptotic time performance of the isEmpty( ), pop( ) and push( ) operations.

2. Using only the operations of the stack given in the class definition (i.e. without using the stack's copy constructor or assignment operator), write a C++ function that returns a copy of the user specified stack. The prototype for the function is given below

```
template< class Object >
Stack< Object > CopyStack( Stack< Object >& stack );
```

Queues

1. Using the operations of the stack and queue, write a function that determines if a string is a palindrome (i.e. reads the same backward and forward; e.g. "level"). The prototype for this function is given below.

```
bool IsPalindrome( const string& theString );
```

2. Suppose that Q is an initially empty array-based queue of size 5. Show the values of the data members front and back after each statement has been executed. Indicate and errors that might occur.

```
Queue< char > Q( 5 );          front = _____   back = _____
Q.enqueue( 'A' );              front = _____   back = _____
Q.enqueue( 'B' );              front = _____   back = _____
Q.enqueue( 'C' );              front = _____   back = _____
char c = Q.dequeue( );         front = _____   back = _____
Q.enqueue( 'A' );              front = _____   back = _____
```

3. Discuss the advantages and disadvantages of the linked list and array-based implementations of a queue.

4. Describe three "real life" applications of a queue.

Dequeue

1. Using only the operations of a deque, write a function that determines if a string is a palindrome (i.e. reads the same backward and forward; e.g. "level"). The prototype for this function is given below.

```
bool IsPalindrome( const string& theString );
```

2. Given an initially empty Deque named D, perform the operations below in the order shown, then answer the questions below

```
Deque< int > D;
D.insertAtBack( 42 );
D.insertAtBack( 17 );
D.insertAtFront( 12 );
D.insertAtBack( 87);
D.removeFromFront( );
D.insertAtBack( 23 );
D.removeFromFront( );
D.insertAtBack( 99 );
D.insertAtFront( 88 );
D.insertAtFront(44 );
D.removeFromBack( );
```

   (a) How many elements are in the deque?

   (b) What value is at the front of the deque?

   (c) What value is at the back of the deque?

3. Explain how to implement a queue using a deque.

4. Explain how to implement a stack using a deque.

5. Explain the programming concept of the "adapter design pattern" as it applies to the implementation of the Deque, Stack, Queue and List.

Definition of the Stack Class
This is the definition of the array based stack from the text .

```
template< class Object >
class Stack
{
    public:
        explicit Stack( int capacity = 10 );
        bool isEmpty( ) const;
        bool isFull ( ) const;
        const Object& top( ) const;
        void makeEmpty( );
        void pop( );
        void push( const Object& x );
        Object topAndPop( ) ;
    private:
        vector< Object > theArray;
        int topOfStack;
};
```

Definition of the Queue Class
This is the definition of the array based queue from the text.

```
template< class Object >
class Queue
{
    public:
        explicit Queue( int capacity = 10 );
        bool isEmpty( ) const;
        bool isFull ( ) const;
        const Object& getFront( ) const;
        void makeEmpty( );
        Object dequeue( );
        void enqueue( const Object& x );
    private:
        vector< Object > theArray;
        int currentSize, front, back;
        void increment( int& x );
};
```

Definition of the Deque Class
This definition is taken directly from the class notes.

```
template < class Object >
class Deque
{
    public:
        Deque( );
        Deque( const Deque& deq );
        ~Deque( );
        bool isEmpty ( ) const;
        void makeEmpty( );
        void insertAtFront( const Object& x );
        void insertAtBack( const Object& x ) ;
        Object removeFromFront( );
        Object removeFromBack( );
        const Deque& operator=( const Deque& rhs );
    private:
        List< Object > theList;
};
```