

# Homework 2 - Due via Electronic Turnin Tuesday, October 26 at 8:00 pm

71 points total

**You must turn in your homework in ps or pdf format**

**To earn 10% extra credit, be sure to include the names of your partners and how long you worked on the homework together. Remember that you must work with at least 2 other people to form a group of three and you must work for at least 2 hours together (minimum of 1 hour sessions).**

1. (6 pts) Show how to implement a queue using two stacks. Describe the Enqueue and Dequeue functions and give the Big-Oh runtime of both. You may assume the Stack has the functions Push, Pop, and Empty. You do not actually have to write code, you may just describe the method and use pictures, but be sure you are clear.

2. (6 pts) Assume that you have a queue implemented with a singly linked list with a head and tail pointers. The queue can be implemented in one of two ways:

Option 1 – Enqueue at tail of list and dequeue at head of list.

Option 2 – Enqueue at head of list and dequeue at tail of list.

Determine which option is a better implementation of a linked list implemented queue based on the runtime of the operations for both options.

3. (10 pts) C-5.12 on page 248 in the textbook. Design algorithms for reversing a sequence that access the sequence only through a restricted set of functions, as indicated below. Each algorithm should rearrange the elements in the sequence. Returning a new sequence is not allowed, although other sequences may be used for auxiliary storage. **Look at the function prototypes in the book and pay special attention to the parameters and return values of each function. You may directly access the element if necessary (i.e. last()->element).**

a) Reverse a sequence using only the functions size(), first(), last(), remove(), and insertFirst().

b) Reverse a sequence using only the functions size(), first(), remove(), and insertFirst().

4. (10 pts) Consider the following classes:

```
class Sphere {  
public:  
    double getArea() const;
```

```

        void displayStatistics () const;
};
class Ball : public Sphere {
public:
    double getArea () const;
    void displayStatistics () const;
};

```

Suppose that the implementation of each version of displayStatistics invokes the function getArea.

a) If mySphere is an instance of Sphere and myBall is an instance of Ball, which version of getArea does each of the following calls to displayStatistics invoke? Explain your answer.

```

mySphere.displayStatistics()
myBall.displayStatistics()

```

b) If the statements

```

Sphere* spherePtr;
Ball* ballPtr;

```

declare spherePtr and ballPtr, which version of getArea does each of the following calls to displayStatistics invoke? Explain your answer.

```

spherePtr->displayStatistics();
spherePtr = &myBall;
spherePtr->displayStatistics();
ballPtr->displayStatistics();

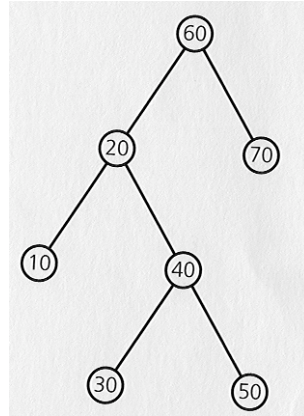
```

5. (5 pts) Write a templated function to determine the minimum of two variables of the template type.

6. (10 pts) Define a class template for an array-based implementation of the ADT stack. Implement the member functions push, pop, top, and empty. (Remember the constructor and destructor)

7. (10 pts) Write a function to perform binary search for a given key in an array of sorted items. What are the best case, worst case, and average case running times for you algorithm?

8. (14 pts) Given the following tree:



What node or nodes are:

- a) The tree's root
- b) Parents
- c) Children of the parents in part b
- d) Siblings
- e) Ancestors of 50
- f) Descendants of 20
- g) Leaves