**Introduction to Computer Science II**

**Lab and Homework 3**

**Due by 1:30pm on Tuesday, January 28**

**Reading**

Read sections 8.1 through 8.6 of the textbook.

**Problems**

**Implement the below functions in**[homework3.py](http://reed.cs.depaul.edu/lperkovic/csc242/homeworks/homework3.py)**and submit through COL.**  
**0.**    Lab attendance is required and constitutes 20% of the lab grade.  
 **1.**   Implement a container class called PriorityQueue that supports methods insert() (a number or other comparable type of objecct),  min() (returns the smallest object in the container), removeMin() (removes and returns the smallest object in the container), as well as overloaded operator len() (returns the size of the priority queue).  
  
In this implementation, you will use a list to store the numbers in the priority queue. Your implementation must insure that the list is sorted after every operation (insertion or removal) so that the minimum number is stored in one end of the list.

Usage:  
>>> pq = PriorityQueue()  
>>> pq.insert(4)  
>>> pq.insert(2)  
>>> pq.insert(7)  
>>> pq.insert(5)  
>>> len(pq)  
4  
>>> pq.min()  
2  
>>> pq.removeMin()  
2  
>>> pq.removeMin()  
4  
>>> pq.removeMin()  
5  
>>> pq.minimum()  
7  
>>> len(pq)  
1

**2.**    Reimplement PriorityQueue (and call it PriorityQueue2) so that the insert() method simply appends the inserted number to the end of the list. Note that in this implementation, the list storing the list items will not be sorted at all times.  
  
  
**3.**    Implement class Hand that represents a hand of playing cards. The class should have a constructor that takes as input the player ID (a string). It should support method addCard() that takes a card as input and adds it to the hand, method showHand() that displays the player’s hand in the format shown, and overloaded funcction len() that returns the number of cards in the hand.

Usage:  
>>> hand = Hand('House')  
>>> deck = Deck()  
>>> deck.shuffle()  
>>> hand.addCard(deck.dealCard())  
>>> hand.addCard(deck.dealCard())  
>>> hand.addCard(deck.dealCard()  
>>> len(hand)  
3  
>>> hand.showHand()  
House:   5 ♡   A ♣   5 ♠

**4.**    Implement class BlackjackHand that represents a blackjack hand. The class should be a sublass of class Hand and inherit its methods. It should also support methods total() that returns the best possible Blackjack value for the hand (you should re-use appropriately the code from function total() on page 201 of the textbook) and method compareHands() that is used to compare one hand to another.

Usage:  
>>> deck = Deck()  
>>> deck.shuffle()  
>>> player = BlackjackHand('Player')  
>>> house = BlackjackHand('House')  
>>> player.addCard(deck.dealCard())  
>>> house.addCard(deck.dealCard())  
>>> player.addCard(deck.dealCard())  
>>> house.addCard(deck.dealCard())  
>>> player.showHand()  
Player:   A ♣   J ♠  
>>> house.showHand()  
House:   A ♡   K ♡  
>>> player.total()  
21  
>>> house.total()

**5.**Implement classes Square and Triangle as subclasses of the given class Polygon. Each will overload the constructor method \_\_init\_\_ so it takes only one argument, the side length, and each will override methodarea() that computes the area using a simpler implementation. The method \_\_init\_\_ should make use of the superclass \_\_init\_\_ method, **so no instance variables (s and n) are defined in subclasses**. Note: The area of an equilateral triangle of side length s is (s2 \* √3) / 4  ; you should know the area of a square of side length s. 

Usage:  
>>> s = Square(2)  
 >>> s.perimeter()  
8   
>>> s.area()   
4   
>>> t = Triangle(3)  
>>> t.perimeter()  
9  
 >>> t.area()  
6.3639610306789285