LAST NAME (PRINT): ____________________ FIRST NAME (PRINT): ____________________

CS 367: Introduction to Data Structures
Midterm Sample Questions

Friday, July 14th 2017.

100 points (26% of final grade)

Instructor: Meena Syamkumar

1. **Fill in these fields and their bubbles on the scantron form (use #2 pencil).**
   (a) LAST NAME - fill in your last (family) name starting at left column.
   (b) FIRST NAME - fill in first five letters of your first (given) name.
   (c) IDENTIFICATION NUMBER is your UW Student ID number.
2. **DOUBLE-CHECK THAT YOU HAVE FILLED IN THE BUBBLES FOR EACH COLUMN OF ABOVE FIELDS ON SCANTRON.**
3. **Read, agree to, and sign this ACADEMIC CONDUCT STATEMENT.**

   I will keep my answers covered so that they may not be viewed by another student during the exam or prior to completion of their exam. I will not view or in any way use another?s work or any unauthorized devices. I understand that I may not make any type of copy of any portion of this exam. I understand that being caught doing any of these or other actions that permit me or another student to submit work that is not our own will result in automatic failure of the course. All such penalties are reported to the Deans Office for all involved.

Signature:

Parts	Number of Questions	Question Format	Possible Points
I	15	Simple Choice	15
II	15	Multiple Choice	45
III	10	Written	40
	35	Total	100

Turn off and put away all electronic devices and wait for the proctor to signal the start of the exam.

Part I: Simple Choice (1 point each)

Select the word or phrase that makes the statement **true**.

If there is no ______________, then mark **A** if the statement is true or **B** if the statement is false. Be sure to mark the corresponding letter on the answer sheet (scantron). Unless otherwise specified, assume the ADTs, data structures, interfaces, and algorithms mentioned are those discussed in lecture and in the readings.

Questions 1-2, assume **StackADT** is implemented using a circular, singly-linked chain of nodes with only a reference to the *head* of the chain, and where the *top* of the stack is at the *head* of the chain. What is the *worst-case* time complexity for the following StackADT operations given a size of N.

1. push(E item) adds a new node containing item to the top of this stack. A. O(1) B. O(N)
2. pop() removes the node at the top of this stack and returns this node's item. A. O(1) B. O(N)

For **questions 3-4**, select the word or phrase that makes the statement **true**. Mark the corresponding letter on the answer sheet (scantron).

3. A reason to design an iterator class to directly access a collection's data structure is because the collection's public interface ______________ support the operations required by the iterator.

   A. does B. does not

4. Coding and using an ADT for a program's variable type ______________ the programmer to choose between different implementations.

   A. allows B. does not allow

Part II: Multiple Choice (3 points each) Choose the best answer of the available choices.

5. Suppose myList is a **List ADT** that initially contains Strings in this order:
   ["A", "B", "C", "D", "E", "F"]

   Consider the following code fragment:

   ```
   for (int i = 0; i < myList.size(); i++)
           myList.add(myList.remove(i));
   ```

   Which one below correctly shows the resulting contents of myList after the code fragment executes?

   A. ["A", "B", "C", "D", "E", "F"]
   B. ["B", "D", "F", "A", "C", "E"]
   C. ["B", "D", "F", "C", "A", "E"]
   D. ["B", "D", "F", "C", "E", "A"]
   E. ["F", "E", "D", "C", "B", "A"]

6. Consider the following code fragment using a **List ADT**:

```
ListADT<Integer> myList = new SimpleArrayList<Integer>();
for (int i = 7; i >= 0; i--)
        myList.add(i);
Iterator<Integer> itr = myList.iterator();
for (int i = 0; i < 10; i++) {
        try {
                System.out.print(itr.next());
        } catch (NoSuchElementException e) {
                System.out.print(" end list");
        }
}
```

Which one of the following correctly shows what is output when the code above is executed?

A. 76543210
B. 01234567 end list
C. 76543210 end list
D. 01234567 end list end list
E. 76543210 end list end list

7. Consider the following method that operates on a **Stack ADT**:

```
public static boolean mystery (StackADT<Object> st, Object ob)
   {
        boolean result = false;
        StackADT<Object> tmpSt = new
           SimpleArrayStack<Object>();
        while (!st.isEmpty() && !result) {
                tmpSt.push(st.pop());
                if (tmpSt.peek().equals(ob))
                        result = true;
        }
        return result;
}
```

Which one of the following best describes what *mystery* does?

A. It returns true if and only if stack st contains at least one occurrence of ob and it restores stack st to its original state.
B. It returns true if and only if stack st contains at least one occurrence of ob but it does not restore stack st to its original state.
C. It returns true if and only if stack st contains more than one occurrence of ob and it restores stack st to its original state.
D. It returns true if and only if stack st does not contain ob and it restores stack st to its original state.
E. It returns true if and only if stack st does not contain ob but it does not restore stack st to its original state.

8. Consider the following incomplete code fragment using a List ADT:

```
ListADT<String> words ... //assume words contains N strings
ListADT<String> rev = new SimpleLinkedList<String>();
for (int i = INIT; COND; EXPR) {
        rev.add(ARG);
}
```

If SimpleLinkedList is implemented with a singly-linked chain of nodes with only a head reference, which one of the following replacements for INIT, COND, EXPR, and ARG completes the code fragment so that it correctly and efficiently creates a copy in reverse order of the List words?

INITialization	**COND**ition	**EXPR**ession	**ARG**ument(s)
A. 0	i < words.size()	i++	words.get(i)
B. 0	i < words.size()	i++	0, words.get(i)
C. words.size() - 1	i < words.size()	i++	0, words.get(i)
D. words.size() - 1	i >= 0	i- -	words.get(i)
E. words.size() - 1	i >= 0	i- -	0, words.get(i)

Part III: Written (4 points each)
Write your answer within the provided space as briefly as possible.

9. Complete the following **recursive** implementation to print the contents of a singly linked list in forward order

```
void print(Listnode<Integer> curr) {
```

10. Complete the following implementation of **insert** function that will insert items into a **Max Heap**

```
private Comparable[] items;
private int numItems;

public void insert(Comparable item) {
```