# CS304 Midterm Practice Exam
# Fall 2016

## Multiple Choice

1. In which situations the array based implementation and the linked based implementation are most suitable?

    A. Unbounded `Stack` ADT and bounded `Stack` ADT, respectively  
    B. Unbounded `Stack` ADT and unbounded `Stack` ADT, respectively  
    C. Bounded `Stack` ADT and bounded `Stack` ADT, respectively  
    **D. Bounded `Stack` ADT and unbounded `Stack` ADT, respectively**  
    E. They are equally suitable.

2. Which of the following represents "quadratic" time?

    **A. $3n+5n^2+1$**  
    B. $n\log n$  
    C. $n^7+n+1$  
    D. $2^n+n^2+5$  
    E. None of the above

3. Which of the following statements is **TRUE** about linked lists?

    A. All the elements in a linked list are always stored contiguously.  
    B. Given the position, removing a node from a linked list is extremely inefficient.  
    C. Linked lists support random access.  
    D. You can traverse a singly linked list from head to tail and from tail to head.  
    **E. Each node in a singly linked list saves data as well as a link to its next node.**

4. Which of the following Java statement attempts to invoke methods that would potentially throw exception(s)?

    A. `throw`  
    B. `raise`  
    C. `catch`  
    **D. `try`**  
    E. None of the above

5. Given the following code snippet, what is the size of the stack `s` after all this code executes?

```
NumberStack s = new LinkedNumberStack();
s.push(10);
s.push(20);
s.push(30);
s.push(40);
s.push(50);
s.pop();
int myInt = s.top();
s.pop();
s.pop();
s.push(myInt);
```

    A. 2  
    **B. 3**  
    C. 4  
    D. 5  
    E. none of the above

6. Which of the following structures did we classify as “non-linear”?

A. array  
B. sorted list  
C. stack  
D. queue  
**E. tree**

7. Given the following code snippet, what is the element at the front of the queue `q` after all this code executes?

```
NumberQueue q = new LinkedNumberQueue();
q.enqueue(10);
q.enqueue(20);
q.enqueue(30);
q.enqueue(40);
q.enqueue(50);
NumberStack s = new LinkedNumberStack();
for (int i = 0; i < 3; i++) {
   s.push(q.dequeue());
}
while (s.size() > 0) {
   q.enqueue(s.pop());
}
```

A. 10  
B. 20  
C. 30  
**D. 40**  
E. 50

8. Under which circumstances should a recursive solution be preferred to an iterative solution?

(1) When the recursive version is more easily understood.  
(2) When the running time is critical.  
(3) When space (memory) is critical.

A. None of them  
**B. Only (1)**  
C. (1) and (2)  
D. (1) and (3)  
E. All of them

9. Which of the following `Queue` method could throw the `QueueOverflowException`?

**A. `enqueue`**  
B. `dequeue`  
C. `isEmpty`  
D. `isFull`  
E. the constructors

10. For the array based **unbounded** `Queue` implementation, what happens if the underlying array is full and an `enqueue` operation is attempted?

A. A `QueueOverflowException` is thrown.  
B. A `QueueUnderflowException` is thrown.  
C. A linked list is created to hold additional elements.  
**D. A bigger array is created and all the elements in the current array are copied into the new array.**  
E. Nothing happens.

## Short Answers

11. Describe the exceptional cases for the array based bounded `Stack` ADT implementation.

**A `StackUnderflowException` would be thrown when `pop` or `top` is attempted on an empty stack.**

**A `StackOverflowException` would be thrown when `push` is attempted on full stack.**

12. What is the output of the following program?

```
public class TestRecursion {
   public static void main(String[] args){
      myRec(10, 16);
   }

   public static void myRec(int a, int b){
      if (a > 2){
         System.out.println("b is " + b);
         myRec(b - 4, a - 2);

      }
      else
         System.out.println("a is " + a);
   }
}
```

**OUTPUT:**

```
b is 16
b is 8
b is 10
b is 2
a is -2
```

## Coding Questions

13. Assume you are given a recursive definition: $\text{myFun}(n) = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$

a) What are the base case and the general case for this definition? You should list them mathematically. For example:

Base case: factorial(0) = 1
General case: factorial($n$) = $n$ * factorial($n$ - 1)

**Base case: $\mathbf{myFun(1) = 1}$**
**General case: $\mathbf{myFun}(n) = \frac{1}{n} + \mathbf{myFun}(n - 1)$**

b) Write a recursive method to calculate myFun($n$) using the first $n$ terms according to your answer in part **a)**. The method should take an `int` parameter $n$ and return a `double` value.

```
double myFun(int n)
{
   if (n == 1)
      return 1.0;
   else
      return 1.0 / n + myFun(n - 1);
}
```

14. Given the following definition of the LNode class, implement the methods "push" and "pop" for the LinkedStack class. You can assume isEmpty() method is already implemented for you. You may throw a StackUnderflowException if needed without defining it.

```
public class LNode
{
   private int m_value;
   private LNode m_link;

   public LNode(int value) {
      m_value = value;
      m_link = null;
   }

   public void setLink(LNode link) {
      m_link = link;
   }
   public LNode getLink() {
      return m_link;
   }
   public void setInfo(int value) {
      m_info = value;
   }
   public int getInfo() {
      return m_value;
   }
}

public class LinkedStack
{
   private LNode topOfStack;

   public LinkedStack()
   {
      topOfStack = null;
   }

   public void push(int v)
   {
      LNode newNode = new LNode(v);
      newNode.setLink(topOfStack);
      topOfStack = newNode;
   }

   public int pop()
   {
      if (isEmpty())
         throw new StackUnderflowException("Pop attempted on empty stack");
      else
      {
         int v = topOfStack.getInfo();
         topOfStack = topOfStack.getLink();
         return v;
      }
   }
}
```