Section A: Recursion (20± points)

- Draw recursion trace
- Write a recursive method in Java

Section B: Sorting and Search Algorithms, and Comparable Objects (20± points)

- Selection Sort
- Quick Sort
- Comparable object

Section C: Algorithm Complexity / Running Time and Big O Notation (20± points)

- Given an algorithm, use big O notation to represent its running time in terms of the input size n (n is the size of the array).
- Rank different algorithms

Section D: Linked List (20± points)

- In **MyLinkedList** class, more methods are added.
  In the client program, show the output.
- How to use Linked List to implement other data structures (concepts)

Section E: Stacks (20± Points)

- Basic Stack operations
- Use a stack to calculate the result of the following postfix operation.
- Give a Java program with Stack operations, show the output.

****

1. Given an unsorted array, use selection sort to sort it. Show your work
2.
   a. Use big O notation to represent each of the following algorithms' running time in terms of the input size N (for array, N is the size of the array; for linked list, N is the number of nodes).

```
1) Xx
2) Xx
3) Xx
4) Xx
5) Xx
```

   b. Rank above functions in terms of the running time from the best to the worst.

3. Use big O notation to represent each of the following methods' running time in terms of the input size N (N is the size of the array)

`public static void method1(int [] list) {`
`  //some Java code`
`}`
	`public static void method2(int [] list) {`
	`  //some Java code`
	`}`

`public static int method3( int [] myArray ) {`
`  //some Java code`
`}`
	`public static void method4(int [] list) {`
	`  //some Java code`
	`}`

4. Stack operations: push, pop, peek

5. In **MyLinkedList** class, three more methods are added: **method1**, **method2**, and **method3**. All other methods are unchanged.
   In the client program, show the output for each **System.out.println().**

```
public class MyLinkedList< E extends Comparable<E> > {
   private Node<E> head;
   private Node<E> tail;
   private int size;

   public MyLinkedList() {
      head = tail = null;
      size = 0;
   }

   public int getSize() {
      return size;
   }

   public E method1() {

       //some Java code

   }

   public E method2(E obj) {

      //some Java code

   }


   public void method3()  {

      //some Java code
   }

   //isEmpty, addFirst, removeFirst, addLast, and traverse method are unchanged.

}//end of class
```

```
//client program

MyLinkedList<String> airportList = new MyLinkedList<String>();

//assume the airpostList contains "TVF" -> "STC" ->"MSP" -> "NYC" from head to tail


//call method1()

System.out.println( ??? );   _____________________________________

//call method2()

System.out.println( ??? ); _____________________________________

//call method3()

System.out.println( ???  );    _____________________________________
```

6. A question related to Linked List and other data structure. You can answer it in English, or pseudocode, or Java code.

7. What is the output of the following section of Java program?

```
String input = "Wild";

//s1, s2, s3 are Stacks of Character.


//in a loop, call push() and/or peek() method for s1


//in a loop, call push() and/or peek() method for s2



//in a loop, call push() and/or peek() method for s3


//in a loop, call pop() method
   System.out.print ( s1.pop() );    // 1. _________________

//in a loop, call pop() method
   System.out.print ( s2.pop() );    // 2. _________________

//in a loop, call pop() method
   System.out.print ( s3.pop() );    // 3. _________________
```