PART I: Problems: (SOLUTIONS)

Problem 1:
Show the content of initially empty stacks stack1 and stack2 after the following operations

```java
code
stack2.push(23)
stack2.push(7)
stack2.push(50)
stack1.push(42)
int top1 = stack1.peek();
int top2 = stack2.peek();
stack1.push(top2);
stack1.push(top1);
stack2.pop();
```

Answer:

```text
stack1: 42, 50
stack2: 7, 23
```

Problem 4:
Given two sorted stacks stack1 and stack2, implement a function mergeStacks that merges the content of stack1 and stack2 in a new stack result_stack in such a way that the result stack is also sorted (either descendinly or ascendingly).

```java
code
public static Stack<Integer> mergeStacks(Stack<Integer> stack1, Stack<Integer> stack2){
    Stack<Integer> result_stack = new Stack<Integer>();
    // A stack peek is stack1.peek()
    // A stack pop is stack1.pop()
    // To check if stack is empty stack1.isEmpty()

    while (!stack1.isEmpty() && !stack2.isEmpty()){ // END WHILE
        if(stack1.peek()<stack2.peek()){
            result_stack.push(stack1.peek());
            stack1.pop();
        }
        else if(stack1.peek()>stack2.peek()){
            result_stack.push(stack2.peek());
            stack2.pop();
        } else{ // This is in case the elements are equal
            result_stack.push(stack2.peek());
            result_stack.push(stack1.peek());
            stack2.pop();
            stack1.pop();
        }
    }
    // You consumed the 2 stacks and now you need to push the remaining elements
    while(!stack2.isEmpty()){
        result_stack.push(stack2.pop())
    }
}
```
while(!stack1.empty()){
    result_stack.push(stack1.pop())
}
return result_stack;

Problem 5:
Given a stack, write a function getMax that finds the max element in the stack and returns it.

```java
public static int getMax(Stack<Integer> stack){
    int max;
    if(stack.isEmpty()) throw new StackException(“Stack is Empty !!”);
    max = stack.peek();
    while(!stack.isEmpty()){
        if(max<stack.peek()){
            max = stack.peek();
        }else{
            stack.pop()
        }
    }
    return max;
}
```

Problem 6:
Given a queue, write a function getMax that finds the max element in the queue and returns it.

```java
public static int getMax(Queue<Integer> queue){
    int max;
    if(queue.isEmpty()) throw new QueueException(“Queue is Empty !!”);
    max = queue.peek();
    while(!queue.isEmpty()){
        if(max<queue.peek()){
            max = queue.peek();
        }else{
            queue.poll()
        }
    }
    return max;
}
```

Problem 7:
Given two stacks stack1 and stack2, write a function isSimilar that checks whether the two stack are exactly similar and returns true if it is the case; otherwise, it return false.

```java
public static boolean isSimilar(Stack<Integer> stack1, Stack<Integer> stack2){
    // A stack pop is    stack1.pop()
    // To check if stack is empty    stack1.isEmpty()

    while (!stack1.isEmpty() && !stack2.isEmpty()){
        if(stack1.pop()!=stack2.pop())
            return false;
    }
    if(stack1.isEmpty() && stack2.isEmpty())
```
return true;
else return false;
}

Problem 8:

a. If the sequence of operations - push (1), push (2), pop, push (1), push (2), pop, pop, push (2), pop are performed on an initially empty stack, the sequence of popped out values is:

Sequence is: ________22112________________________

b. If the sequence of operations – push (2), pop, push (1), push (2), pop, push(6) popAll are performed on an initially empty stack, the sequence of popped out values is:

Sequence is: ________2261________________________

Problem 9:
What is the output of the following code?

```java
int [] values = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19 };
Stack<Integer> s = new Stack<Integer>();
for (int i = 0; i < values.length; i++){
    s.push( values[i] );
}
int n = 25;
for (int i = 0; i < 4; i++){
    n += s.pop( );
}
for (int i = 0; i < 2; i++){
    n -= s.pop( );
}
System.out.println( n );
```

Output: 69

Problem 10: Suppose that Q is an initially empty circular array-based queue of size 5. Show the values of the front and back after each statement has been executed

ArrayQueue<Character> Q = new ArrayQueue<Character>(5);
front = __0__ back = __4__
Q.enqueue ( 'A' );
front = __0__ back = __0__
Q.enqueue ( 'B' );
front = __0__ back = __1__
Q.enqueue ( 'C' );
front = __0__ back = __2__
char c = Q.dequeue();
front = __1__ back = __2__
Q.enqueue ( c );
front = __1__ back = __3__
Q.peek();
front = __1__ back = __3__

Problem 11:
Suppose that s is an initially empty array-based stack of size 5. Show the values of the top after each statement has been executed.

ArrayStack<Character> s = new ArrayStack<Character>(5);
top = __-1__
Problem 12:
Show the content of initially empty queues `queue1` and `queue2` after the following operations (from front to back order):

```
s.push('A'); top = __0__
s.push('B'); top = __1__
s.push('C'); top = __2__
char c = s.pop();
s.push(c);
```

Answer:
```
queue1
   back   front
      42  42
queue2
   back   front
      23  50  7
```

Problem 13:
Give two reasons why would you use an ArrayList instead of an array to implement a stack or a queue?

- **ArrayLists are dynamic data structures that does not require the knowledge of the structure size prior to runtime.**
- **ArrayList do not throw a stack/queue overflow exceptions.**

Problem 14:
Consider the linked list of integers represented by the following diagram:

```
head
  ● → 5 ● → 3 ● → 7 ● → 12 ● → 10
```

Draw a diagram of the above list after the following lines of code have been executed:

```
Node cur = head.next;
Node nodeToInsert = new Node(4);
nodeToInsert.next = cur.next;
cur.next = nodeToInsert;
```
Problem 15:
For each of the following scenarios choose the “best” data structure from the following list of data structures: an array, linked list, stack, queue. In each case, justify your answer briefly.

a. Suppose that a grocery store decided that customers who come first will be served first
   Answer: Queue

b. A list must be maintained so that any element can be accessed randomly.
   Answer: Array

c. A program needs to remember operations it performed in opposite order.
   Answer: Stack

d. The size of a file is unknown. The entries need to be entered as they come in. Entries must be deleted when they are no longer needed. It is important that structure has flexible memory management
   Answer: Linked List

Problem 16:
Given the following linked list, write the code to perform the following operations:

a. Create a new node (toBeInserted) having value ‘4’ and insert it immediately after ‘5’

   ```java
   Node toBeInserted = new Node(4);
   toBeInserted.next = cur.next;
   cur.next = nodeToInsert;
   ```

b. Delete the head node

   ```java
   head = head.next;
   ```

Problem 17:
In each part below, you are given a segment of java code. Draw a picture that shows the final result of the execution of the code segment. Your picture should indicate the value of every declared variable and the value of every field in every node.

Example: If the code segment is:

   ```java
   Node p = new Node(5);
   p.next = null;
   ```

   The picture should look like this:
Problem 18:

Assume a linked list that holds integers. Suppose that you want to count how many positive values are on the list. Write a function `countPositive` that returns the count of values.

```java
public static int countPositive(Node head){
    Node cur;
    int count=0;
    // INSERT CODE HERE
    for(Node cur = head; cur!=null ; cur = cur.next){
        if(cur.item>=0)
            count++;
    }
    return count;
}
```

Problem 19:

Assume a linked list that holds at least 2 integers. Suppose that you want to insert a new node in the end of the list ONLY if its value is greater than the value of the last node of the list. Write a function `insertWithCondition` that performs the insertions.

```java
public static void insertWithCondition(Node head, Node toBeInserted){
    Node prev, cur;
    for(prev = null, cur = head; cur!=null ; prev= cur, cur = cur.next){
    }
    If(prev.item< toBeInserted.item){
```
prev.next = toBeInserted; 
}
}

**Problem 20:**

Write a function encryptList that traverse a linked list and prints double the data value of the nodes

```java
public static void encryptList(Node head) {
    Node cur;
    for (cur = head; cur != null; cur = cur.next) {
        System.out.println(2 * cur.item)
    }
}
```

**Problem 21:**

Identify the 3 errors in the following table that represents a postfix to infix transformation. Consider the following postfix expression

```
5 9 3 + 4 2 * * 7 + *
```

Here is a chain of operations

<table>
<thead>
<tr>
<th>Stack Operations</th>
<th>Current Stack</th>
<th>Problem ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>push(5);</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>push(9);</td>
<td>5 9</td>
<td></td>
</tr>
<tr>
<td>push(3);</td>
<td>5 9 3</td>
<td></td>
</tr>
<tr>
<td>push(pop() + pop())</td>
<td>9 8</td>
<td>Stack should be 5 12</td>
</tr>
<tr>
<td>push(4);</td>
<td>9 8 4</td>
<td></td>
</tr>
<tr>
<td>push(2);</td>
<td>9 8 4 2</td>
<td></td>
</tr>
<tr>
<td>push(pop() * pop())</td>
<td>9 2 24</td>
<td>Stack should be 9 8 8</td>
</tr>
<tr>
<td>push(pop() * pop())</td>
<td>24 18</td>
<td>Stack should be 9 48</td>
</tr>
<tr>
<td>push(7)</td>
<td>24 18 7</td>
<td></td>
</tr>
<tr>
<td>push(pop() + pop())</td>
<td>24 25</td>
<td></td>
</tr>
<tr>
<td>push(pop() * pop())</td>
<td>600</td>
<td></td>
</tr>
</tbody>
</table>

**Problem 22:**

In each part below, you are given a segment of java code. Draw a picture that shows the final result of the execution of the code segment. Your picture should indicate the value of arrayList representing the queue.

**Example:** If the code segment is:
```java
ReferencedBasedQueue q = new ReferencedBasedQueue();
q.enqueue(5);
q.enqueue(3);
```

```java
ReferencedBasedQueue q = new ReferencedBasedQueue();
ReferencedBasedQueue p = q;
q.enqueue(5);
p.enqueue(3);
q.enqueue(2);
p.dequeue();
```
ReferencedBasedQueue q = new ReferencedBasedQueue();
q.enqueue(5);
q.enqueue(3);
q.enqueue(2);
q.dequeue();

Problem 23

In each part below, you are given a segment of Java code. Draw a picture that shows the final result of the execution of the code segment. Your picture should indicate the value of ArrayList representing the stack.

Example: If the code segment is:
ReferencedBasedStack s = new ReferencedBasedStack();
s.push(5);
s.push(3);

ReferencedBasedStack s = new ReferencedBasedStack();
ReferencedBasedStack c = s;
c.push(5);
s.push(3);
s.push(2);
c.pop();

ReferencedBasedStack s = new ReferencedBasedStack();
s.push(5);
s.push(3);
s.push(2);
s.popAll();
s.push(0);

ReferencedBasedStack s1 = new ReferencedBasedStack();
ReferencedBasedStack s2 = new ReferencedBasedStack();
s1.push(5);
s2.push(3);
s1.push(2);
s2.popAll();
s1.push(0);
ReferencedBasedStack s1 = new ReferencedBasedStack();
ReferencedBasedStack s2 = s1;
s1.push(5);
s2.push(3);
s1.push(2);
s2.popAll();
s1.push(0);

Problem 24
Write a queue implementation using linked lists.

```java
public class LinkedListBasedQueue implements QueueInterface{
    Node head;

    public LinkedListBasedQueue(){
        head = null;
    }

    public boolean isEmpty() {
        if(head==null) return true;
        return false;
    }

    public void enqueue(int newItem) {  // Insert at the end of list
        Node prev, cur;
        Node toInsert = new Node(newItem);
        if(isEmpty()){  
            head = toInsert
        } else{
            for(prev=null, cur= head;cur!=null;prev=cur, cur=cur.next){
                prev.next = toInsert;
            }
        }
    }

    public int dequeue() throws QueueException { // Remove 1st element from the list
        int popped;
        if(isEmpty()){
            throw new QueueException("Can not dequeue on empty queue")
        } else{
            popped = head.item;
            head = head.next;
        }
        return popped;
    }

    public void dequeueAll() {
        head = null;
    }

    public int peek() throws QueueException {
```
int peek;
    If(isEmpty()){
        Throw new QueueException("Can not peek on empty queue")
    }
    else{
        peek = head.item;
    }
    return peek;