Midterm Exam Study Guide 😊

1. Know the Stack array-based implementations (how the top counter changes with push/pop operations).
2. Know that Stacks can be used to implement recursive functions.
3. Know the Queue circular array-based and implementation (how the back and front changes with enqueue/dequeue operations)
4. Know the Queue reference-based implementations
5. Know the Stack reference-based implementations
6. Know the advantages and disadvantages of array based and reference based implementations.
7. Master the Infix to Postfix and the Postfix to Infix algorithms. (Youtube has excellent videos in case you are still struggling) (Make sure how the modulus (%) operation works 😊)

→ 2 % 10 = 2  1 % 10 = 1  10 % 1 = 0  10 % 2 = 0

8. Review all the labs covered from Lab2 to Lab5
9. Know all the linked list operations including the special case ones (insert/delete at the beginning, insert/delete at the end)
10. Read carefully all the slides covered in class and ask me on Monday in case of any doubt !!!!
Midterm Exam I: Review Problems

PART I: Problems:

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

```java
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:

```
stack1
stack2
```

Problem 2:

Use the postfix to infix algorithm to transform the following expressions into its infix form, then evaluate the expression following values for identifiers: A=8, B=7, C=5, D=2, and E=1. Show the content of the stack each time you scan a new token from the expression:

1. \( A \ B \ - \ C \ * \ D \ E \ + \ / \)

![Diagram](attachment:diagram.png)

Final Infix Expression: __________________________
Final Result: __________________________
2. \( A \ D + C / B \ E ^ \ + \)

3. \( C \ E A ^ \ / B \ D - \)
4. \( B \ C + \ D / \ E \ A ^ \ / \)

5. \( A \ B - C \% \ D \ E / \ E / \ A \ * \)

6. \( A \ C \ D + / \ B / \ E \ % \)

7. \( A \ D + C + C \% \ B \ C - * \)
Problem 3:

Use the infix to postfix algorithm to transform the following expressions into its postfix form. Show the content of the stack and postfix list each time you scan a new token from the expression:

1. \((A + C / D ^ E ) + B\)

2. \((B \% C) - D ^ E + B\)
3. \[ A - B \% ( C + D \^ E ) \]

<table>
<thead>
<tr>
<th>Pf:</th>
<th>Pf:</th>
<th>Pf:</th>
<th>Pf:</th>
<th>Pf:</th>
<th>Pf:</th>
</tr>
</thead>
</table>

Final Postfix Expression: ______________________

4. \[ E - ( A + B ) \^ C / D \]

<table>
<thead>
<tr>
<th>Pf:</th>
<th>Pf:</th>
<th>Pf:</th>
<th>Pf:</th>
<th>Pf:</th>
<th>Pf:</th>
</tr>
</thead>
</table>

Final Postfix Expression: ______________________

5. \[ ( A + B - C ) \% E - D + ( A / E ) \]
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 descenдинly or ascendingly).

```java
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()
    }
```

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;
    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;

    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
    // To check if stack is empty

    return stack1.isEmpty() && stack2.isEmpty();
}
```

Problem 8:

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

   Sequence is: ______________________________________

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: _______________________________________
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:

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.

```java
ArrayQueue<Character> Q = new ArrayQueue<Character>( 5 );
front = _____  back = _____
Q.enqueue ( 'A' );  front = _____  back = _____
Q.enqueue ( 'B' );  front = _____  back = _____
Q.enqueue ( 'C' );  front = _____  back = _____
char c = Q.dequeue( );  front = _____  back = _____
Q.enqueue ( c );  front = _____  back = _____
Q.peek();  front = _____  back = _____
```

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.

```java
ArrayStack<Character> s = new ArrayStack<Character>( 5 );
top = _____
s.push( 'A' );
top = _____
s.push( 'B' );
top = _____
s.push( 'C' );
top = _____
char c = s.pop( );
top = _____
s.push( c );
top = _____
s.peek();
top = _____
```
Problem 12:

Show the content of initially empty queues queue1 and queue2 after the following operations (from front to back order):

```java
queue2.enqueue(23)
queue2.enqueue(7)
queue2.enqueue(50)
queue1.enqueue(42)
int top1 = queue1.peek()
int top2 = queue2.peek()
queue2.enqueue(top2)
queue1.enqueue(top1)
queue2.dequeue()
```

Answer:

```
queue1: 42, 23, 7, 50
queue2: 7, 23, 50
```

Problem 13:

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

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:

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

Answer:

```
head → 4
```

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

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

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

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’

b. Delete the head node

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:
Node p, q;
p = new Node(5);
p.next = null;
q = new Node(10);
q.next = p;
p.next = p;

Node p, q, r;
p = new Node(5);
p.next = null;
q = new Node(10);
q.next = null;
p.next = q;
r = p;

Node p, q;
p = new Node(5);
p.next = null;
q = new Node(10);
p.next = q;
p.next = q.next;

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

    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;
}
```

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;
}
```

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></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></td>
</tr>
<tr>
<td>push(pop() * pop())</td>
<td>24 18</td>
<td></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);
```

![Queue Diagram](attachment:image.png)

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

![Queue Diagram](attachment:image.png)

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

![Queue Diagram](attachment:image.png)

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:
```java
ReferencedBasedStack s = new ReferencedBasedStack();
s.push(5);
s.push(3);
```

![Stack Diagram](attachment:image.png)

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

![Stack Diagram](attachment:image.png)
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 = s;
s1.push(5);
s2.push(3);
s1.push(2);
s2.popAll();
s1.push(0);

Problem 24
Write a queue implementation using linked lists.

public class LinkedListBasedQueue implements QueueInterface{
    Node head;

    public LinkedListBasedQueue(){

    }

    public boolean isEmpty() {

    }

    public void enqueue(int newItem) {. // Insert at the end of list
public int dequeue() throws QueueException { // Remove 1st element from the list
    int popped;

    return popped;
}

public void dequeueAll() {

}

public int peek() throws QueueException {

}

PART II: Multiple Choice Questions:

1. Consider the algorithm for determining whether a sequence of parentheses is balanced. The maximum number of parentheses that appear on the stack AT ANY TIME when the algorithm analyzes:“()(()(()))” are:
   a) 1
   b) 2
   c) 3
   d) 4

2. Which data structure is used for implementing recursion?
   a) Queue
   b) Stack
   c) Array
   d) List

3. Which data structure is needed to convert infix notation to postfix notation?
   a) Branch
   b) Tree
   c) Queue
   d) Stack
4. If the elements “A”, “B”, “C” and “D” are placed in a queue and are deleted one at a time, in what order will they be removed?
a) ABCD 
b) DCBA 
c) DCAB 
d) ABDC 

5. What is the time complexity of enqueue operation?
a) O(logn) 
b) O(nlogn) 
c) O(n) 
d) O(1) 

6. What is the term for inserting into a full queue known as?
a) overflow 
b) underflow 
c) null pointer exception 
d) all of the mentioned 

7. What does the following piece of code do?

```java
public int function()
{
    if(isEmpty())
        throw new Exception("ERROR!!");
    else
    {
        int high;
        high = q[front];
        return high;
    }
}
```

a) Dequeue 
b) Enqueue 
c) Return the front element 
d) None of the mentioned 

8. In a circular queue, how do you increment the back end of the queue?
a) back++ 
b) (back+1) % MAX_SIZE 
c) (back % MAX_SIZE)+1 
d) back— 

9. Which of the following statement(s) about stack data structure is/are NOT correct?
a) Arrays can be used for implementing Stacks 
b) Top of the Stack always contain the new pushed node
c) Stack is the FIFO data structure
d) ArrayLists can be used for implementing Stacks

10. Consider the following operation performed on a stack of size 5.
Push(1);
Pop();
Push(2);
Push(3);
Pop();
Push(4);
Pop();
Pop();
Push(5);

After the completion of all operation, the number of elements present in stack are
a) 1
b) 2
c) 3
d) 4

11. Which of the following real world scenarios would you associate with a stack data structure?
   a) piling up of chairs one above the other
   b) people standing in a line to be serviced at a counter
   c) offer services based on the priority of the customer
   d) all of the mentioned

12. Which of the following is true of stacks and queues?
   a) A stack is a last-in, first-out structure, and a queue is a first-in, first-out structure
   b) A stack is a first-in, first-out structure, and both structures are random access structures.
   c) A stack is a last-in, first-out structure, and a queue is a random access structure.
   d) A queue is a last-in, first-out structure, and a stack is a first-in, first-out structure.
   e) A queue is a first-in, first-out structure, and a stack is a random access structure.

13. Suppose cursor points to a node in a linked list (using the node definition with member functions called
data and link). What Boolean expression will be true when cursor points to the tail node of the list?
   a) A. (cur == NULL)
   b) B. (cur.next.next == NULL)
   c) C. (cur.item == NULL)
   d) D. (cur.item == 0.0)
   e) E. None of the above.
14. Linked list is considered as an example of __________ type of memory allocation.
   a) Dynamic
   b) Static
   c) Compile time
   d) None of the mentioned

15. What is the functionality of the following piece of code?

```java
public int function(int data, Node head)
{
    Node cur = head;
    int var = 0;
    while(cur != null)
    {
        if(cur.item == data)
        {
            return var;
        }
        var = var + 1;
        cur = cur.next;
    }
    return -1;
}
```

a) Find and delete a given element in the list
b) Find and return the given element in the list
c) Find and return the position of the given element in the list
d) Find and insert a new element in the list

16. How do you insert an element at the beginning of the list?
   a) 
   ```java
   public void insertBegin(Node node, Node head)
   {
       node.next = head;
       head = node;
   }
   ```
   
   b) 
   ```java
   public void insertBegin(Node node, Node head)
   {
       head = node;
       node.next = head;
   }
   ```
   
   c) 
   ```java
   public void insertBegin(Node node, Node head)
   {
       Node temp = head.next;
       node.next = tmp;
       head = node;
   }
   ```
17. Given the code fragment, which of the following expressions has the value null?

```java
Node p = new Node(12);
Node q = new Node(5);
p.next = null;
q.next = p;
```

a) p
b) q
c) q.next
d) q.next.next
e) none of the above

18. Consider the following pseudocode:

```
disable a stack of characters
while ( there are more characters in the word to read )
{
    read a character
    push the character on the stack
}
while ( the stack is not empty )
{
    pop a character off the stack
    write the popped character to the screen
}
```

What is written to the screen for the input “carpets”?

a) serc
b) carpets
c) steprac
d) ccaarrppeetts

19. Assume that head is pointing to the first node of a linked list of many nodes. Which code segment below searches the linked list for the first occurrence of searchVal, leaving cur pointing to the node where it was found? (Assume searchVal is definitely in the list.)

A) Node cur = head;
   while (item != val)
      cur = next;
B) cur = head;
   while (cur.item != searchVal)
      cur = cur.next;
C) cur = head;
   while (cur.item != searchVal)
      cur++;

Justification: __________________
20. Parentheses are not mandatory for either the Infix or Postfix
a) True
b) False
c) Maybe

21. Assume a stack is implemented using a linked list. It is considered a
a) Array-based implementation
b) Reference-based implementation
c) All of the Above

22. Finding the max element in an unordered stack would require
a) O(1) operations
b) O(log n) operations
c) O(n) operations.
d) None of the above

23. Why do we need an additional prev pointer in addition to cur when performing a linked list insertion or deletion?
 a) To be able to perform the operation in O(n) time
b) To be able to update the references of the linked list
c) To have a cool data structure
d) None of the above

24. Why do we call data structures as Abstract Data Type (ADT)?
 a) They have an array-based implementation
b) Their implementation is hidden/abstracted to the user and only the operations on ADTs matter
c) They implement an abstract interface
d) None of the above

25. Given the postfix expression 6 3 + 8 7 − * what is the value of the expression after evaluation in its infix form
a) An integer between -5 and 5
b) An integer between 5 and 10
c) An integer between 10 and 15
d) None of the above
26. Which of the following data structure might give an overflow

a) ArrayList implementation of Stack  
b) Circular array based implementation of Stack  
c) Reference Based implementation of Queue and Stack  
d) All of the above

27. Suppose we have a circular array implementation of a queue with 5 items. the front = 0 and back = 4. The max capacity of the array is MAX_SIZE = 40.

(a) What will the peek operation return to the user
   a) array[front]  
   b) array[front+1]  
   c) array[back]  
   d) None of the Above

(b) What will be the value of front and back after 2 dequeue operations
   a) front = 1   back = 3  
   b) front = 0   back = 2  
   c) front = 2   back = 4  
   d) front = 40   back = 2

(c) Where does the push place the new item:
   a) array[5]  
   b) array[6]  
   c) array[0]  
   d) Queue Overflow

28. One difference between a queue and a stack is:

a) Stacks require dynamic memory, but queues do not.  
b) Queues use two ends of the structure; stacks use only one.  
c) Stacks use two ends of the structure, queues use only one.  
d) Queues use two ends of the structure, stacks use only one.