CSc 2720
Data Structures
Midterm

Spring 2019

Total Points: 100
Duration: 75 minutes

Instructions:

1. This is a closed book and closed notes exam, except for a 8.5x11 single-sided "cheat sheet".
2. Please turn off your cell phone and laptop.
3. Please use the white space provided below each question to answer the question.

Student Name: ____KEY________________________________

Seat Row #: ________ **Position #:** ________

For instructor use only:

Problem	Points
1	/30
2	/15
3	/20
4	/10
5	/10
6	/15
7 (Bonus)	/10
Total	

"That which does not kill us, makes us stronger." - **Friedrich Nietzsche**

Part I: Multiple Choice Questions (30 points: 2 points every question):
For each of the multiple choice questions below, please select the right answer (**only one**).

1. What data structure can be used to check if a syntax has balanced parenthesis?
   a) queue
   b) Stack
   c) Linked List
   d) Tree
   e) None of the above

2. What is the need for a **circular** array compared to a non-circular array-based implementation of a queue?
   a) Effective usage of memory
   b) Easier computations
   c) All of the mentioned
   d) None of the above

3. Which of the following piece of code has the functionality of counting the number of elements in the list?
   a)

```
public int length(Node head){
        int size = 0;
        Node cur = head;
        while(cur!=null){
                size++;
                cur = cur.next;}
        return size;
}
```

b)

```
public int length(Node head){
        int size = 0;
        Node cur = head;
        while(cur!=null){
                cur = cur.item;
                size++;}
        return size;
}
```

c)

```
public int length(Node head){
        int size = 0;
        Node cur = head;
        while(cur!=null){
                size++;
                cur = cur.next;}
}
```

d)

```
public int length(Node head){
        int size = 0;
        Node cur = head;
        while(cur!=null){
                size++;
                cur = cur.next.next;}
        return size;
}
```

4. What is the value of the postfix expression 6 3 2 1 + – * is:
   a) between -5 and -15
   b) between 5 and -5
   c) between 5 and 15
   d) between 15 and 10
   e) None of the above

5. 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:
   a) 2216
   b) 2162
   c) 2261
   d) 2126

6. If the sequence of operations - push (5), push (0), pop, push (9), push (2), pop, pop, pop, push (2), pop are performed on an initially empty stack, the sequence of popped out values is:
   a) 50922
   b) 02952
   c) 09225
   d) 09252

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

```
public int function(int data){
     Node tmp = head;
     int var = 0;
     while(tmp != null){
              if(tmp.item == data){
                       return var;
                       }
              var++;
              tmp = tmp.next;
     }
     return -9999;
}
```

   a) Find and delete a given element in the list if it exists otherwise return -9999
   b) Find and return the given element in the list otherwise return -9999
   c) Find and return the position of the given element in the list otherwise return -9999
   d) Find and insert a new element in the list

8. Linked list and ArrayList data structures are considered as an example of ____________ type of memory allocation.
   a) Dynamic
   b) Static
   c) Compile time
   d) None of the mentioned.

9. Which of the following real world scenarios would you associate with a **queue** 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

10. Given the code fragment. which of the following expressions has the value null?

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

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

11. Consider the following operation performed on a stack of size 2.

```
push(1);
pop();
pop();
push(2);
push(3);
push(4);
pop();
pop();
push(5);
```

How many exceptions are thrown?

a) 1
b) 2
c) 3
d) 4

12. Suppose we have a circular array implementation of a queue with 2 items. the front = 0 and back = 1. The max capacity of the array is MAX_SIZE = 2.

(a) What will the peek operation return to the user

a) array[0]
b) array[1]
c) array[2]
d) Throw QueueException
e) None of the Above

(b) What will be the value of front and back after a dequeue operations

a) front = 1 back = 1
b) front = 1 back = 2
c) front = 0 back = 2
d) front = 0 back = 1

(c) Where does the push place the new item:

a) array[1]
b) array[2]
c) array[0]
d) **Queue Overflow**

13. Which of the following data structure might have an overflow

a) ArrayList implementation of Stack
b) Reference Based implementation of Queue and Stack
c) All of the above
d) **None of the above**

Part II: Problems:
Problem 2 (15 points):

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

ArrayQueue<Character> Q = new ArrayQueue<Character> (10);	front = _**0**___	back = __**9**__
Q.enqueue (‘D’);	front = __**0**__	back = __**0**__
char c = Q.dequeue( );	front = __**1**__	back = __**0**__
Q.enqueue (‘F’);	front = __**1**__	back = __**1**__
Q.enqueue (‘C’);	front = __**1**__	back = __**2**__
Q.enqueue (c);	front = __**1**__	back = __**3**__
Q.peek( );	front = __**1**__	back = __**3**__
Q.dequeue( );	front = __**2**__	back = __**3**__

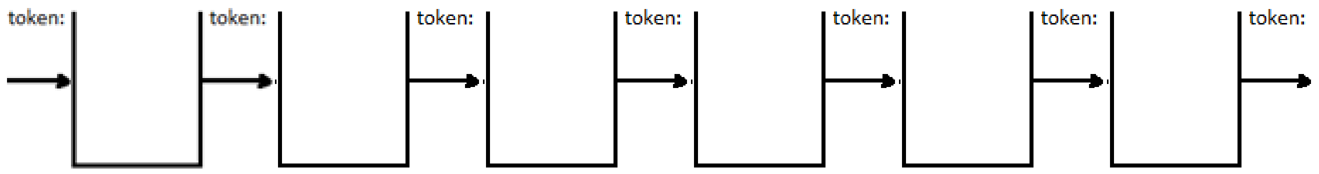
B. Suppose that s is an initially empty array-based stack of size 10. Show the values of the top after each statement has been executed.

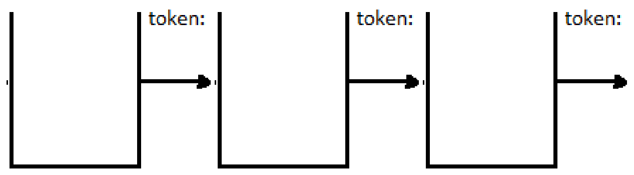
ArrayStack<Character> s = new ArrayStack<Character>(10);	top = __**-1**__
s.push(‘A’);	top = __**0**__
char c = s.pop( );	top = __**-1**__
s.peek( );	top = __**-1**__
s.push(‘D’);	top = __**0**__
s.push(‘C’);	top = __**1**__
s.push(c);	top = __**2**__
s.peek ();	top = __**2**__

Problem 3 (20 points – 5 points each):

A. 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=3, and E=1. Show the content of the stack each time you scan a new token from the expression:

- B D + C / E ^ A *

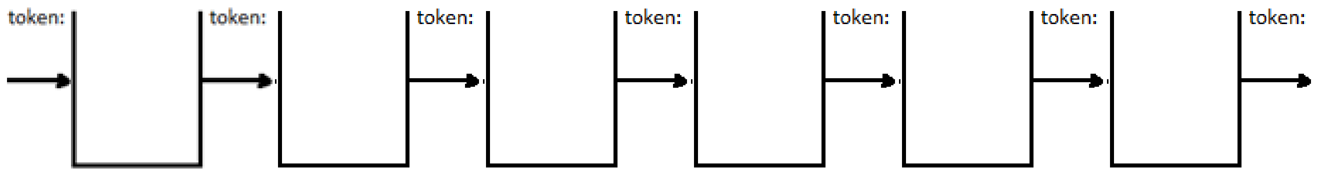


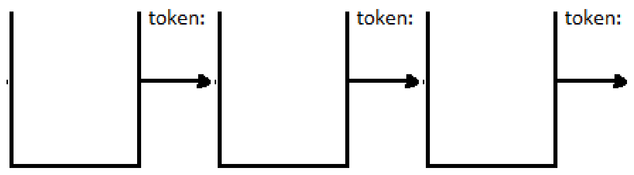


Final Infix Expression : (((B + D) / C) ^ E) * A

Final Result : 16

- E D + B C - * A /



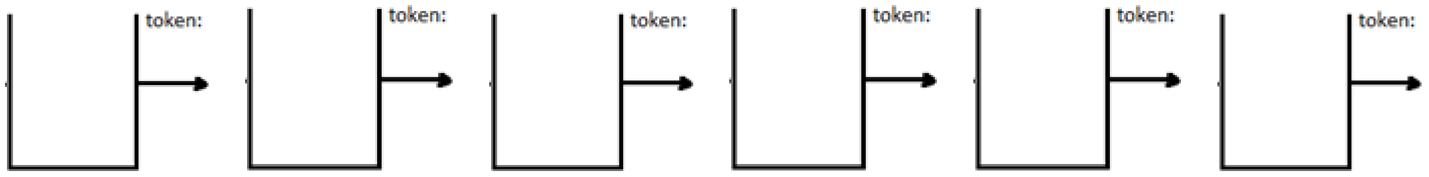


Final Infix Expression : ((E + D) * (B - C)) / A

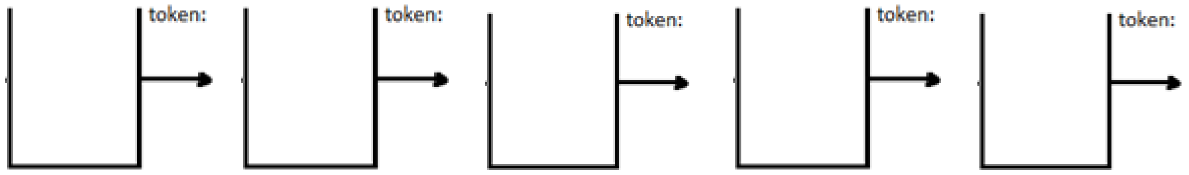
Final Result : 1

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

- (A + C) / D ^ E * B



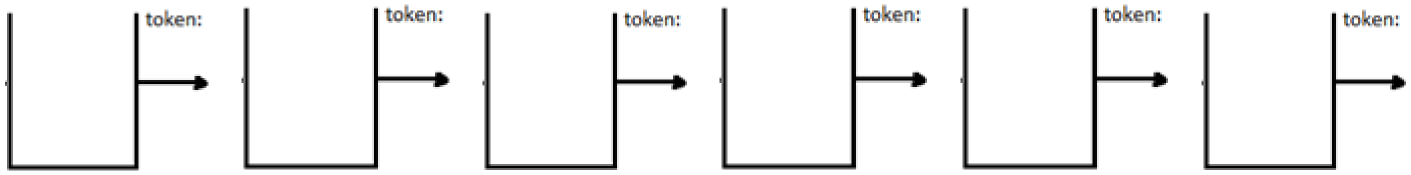
Pf:__________ Pf:__________ Pf:__________ Pf:__________ Pf:__________ Pf:__________



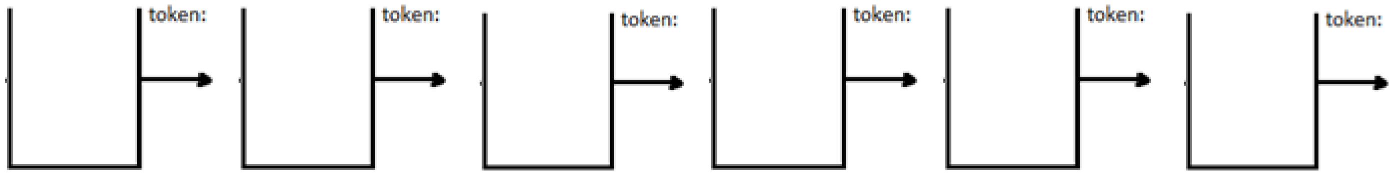
Pf:__________ Pf:__________ Pf:__________ Pf:__________ Pf:__________

Final Postfix Expression : ___ A C + D E ^ / B * ___

- B + E – (A + B) ^ C / D



Pf:__________ Pf:__________ Pf:__________ Pf:__________ Pf:__________ Pf:__________

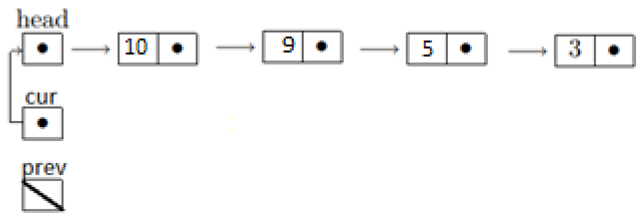


Pf:__________ Pf:__________ Pf:__________ Pf:__________ Pf:__________ Pf:__________

Final Postfix Expression : _ B E + A B + C ^ D / -

Problem 4 (10 points):

Given the following **sorted** linked list, write **the code** to perform the following operations:



a. Create a new node (toBeInserted) having value '4' and, find the right position to insert it (between 5 and 3) and insert in the sorted list.

```
Node prev, cur;
Node toBeInserted = new Node(4);
for(cur=head, prev=null;   cur!=null   ;prev=cur,cur=cur.next){
        if(cur.item <= toBeInserted.item){
                toBeInserted.next = cur;
                prev.next  = toBeInserted;
        }
}
```

b. Delete the last node in the linked list (that has value 3)

```
Node prev, cur;
for(cur=head, prev=null;   cur!=null   ;prev=cur,cur=cur.next){
        }
prev.next = null;
```

c. Delete the first node in the linked list (that has value 10)

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

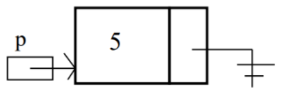
d. Traverse the linked list and count the number of elements in the list that are greater than 5.

```
Node cur;
int count = 0 ;
for(cur=head;   cur!=null   ; cur=cur.next){
        if(cur.item >5){
                count++;
        }
}
```

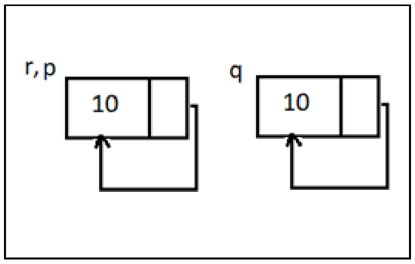
Problem 5 (10 points):
In each part below, you are given a segment of java code. Draw a picture that shows the final result of the execution of the code segment. Your picture should indicate the value of every declared variable and the value of every field in every node.

Example: If the code segment is:

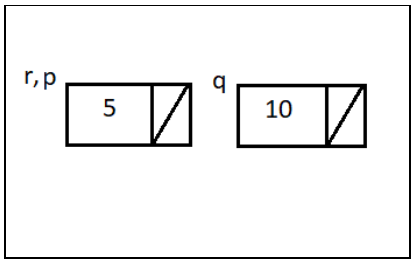
```
Node p = new Node(5);        Picture:
p.next = null;
```



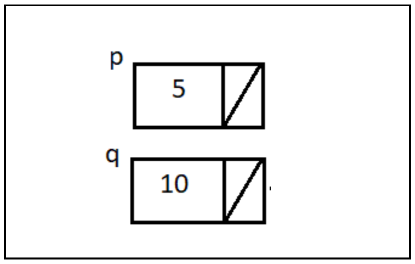
```
Node r, p, q;
p = new Node(10);              Picture:
q = new Node(10);
q.next = q;
p.next = p;
r = p;
```



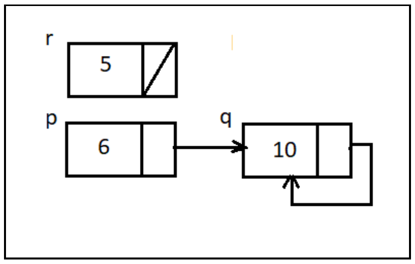
```
 Node p, q, r;
p= new Node(5);                Picture:
p.next = null;
q= new Node(10);
q.next = null;
p.next = q;
r = p;
r.next = null;
```



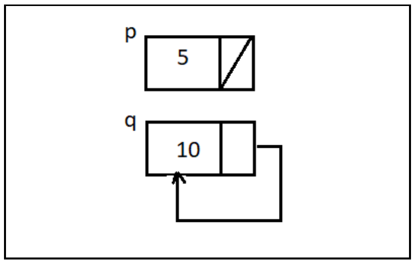
```
Node p, q;
p= new Node(5);                Picture:
p.next = null;
q= new Node(10);
p.next = q;
p.next = q.next;
```



```
 Node r, p, q;
p= new Node(6);
r= new Node(5);                Picture:
p.next = r;
q= new Node(10);
p.next = q;
q.next = p.next;
```



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



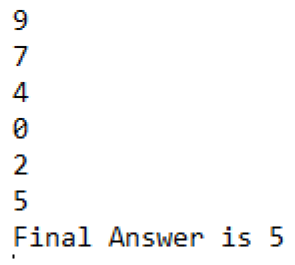
Problem 6 (15 points):
What is the output of the following code snippets?

```
int [] values = {1, 2, 3, 4, 1, 2, 3, 4, 1, 2 };
Queue<Integer> s = new LinkedList<Integer>();
for (int i = 0; i < values.length; i++){
      q.add( values[i] );
}
int n = 10;
for (int i = 0; i < 4; i++){
      n = n - q.poll();
      System.out.println( n );

 }
for (int i = 0; i < 2; i++){
      n =  n + ( q.poll() + 1);
      System.out.println( n );

}
System.out.println(“Final Answer is” +  n );
```

Output:



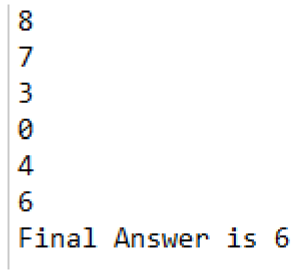
```
9
7
4
0
2
5
Final Answer is 5
```

```
int [] values = {1, 2, 3, 4, 1, 2, 3, 4, 1, 2 };
Stack<Integer> s = new Stack<Integer>();
for (int i = 0; i < values.length; i++){
      s.push( values[i] );
}
int n = 10;
for (int i = 0; i < 4; i++){
      n = n - s.pop();
      System.out.println( n );

 }
for (int i = 0; i < 2; i++){
      n =  n + (s.pop() * 2);
      System.out.println( n );

}
System.out.println(“Final Answer is” +  n );
```

Output:



```
8
7
3
0
4
6
Final Answer is 6
```

Problem 7 (10 points BONUS)
Write a **stack** implementation using **linked lists**.

```
public class LinkedListBasedStack implements StackInterface{
      Node head;

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

      public void push(int newItem) {  // insert at the beginning of the list
            Node prev, cur;
            Node toInsert = new Node(newItem);
                  If(isEmpty()){
                        head = toInsert;
                  }
                  else{
                        toInsert = head.next;
                        head = toInsert;

                  }
      }
      public int pop() throws StackException { // Remove 1st element in the list
            int poped;
            If(isEmpty()){
                        Throw new StackException(“Can not pop on empty stack”)
                  }
                  else{
                        poped = head.item;
                        head = head.next;
                        }

            return poped;
      }
      public void popAll() {
            head = null;

      }
      public int peek() throws StackException {
            int peek;
                  If(isEmpty()){
                        Throw new StackeException(“Can not peek on empty stack”)
                  }
                  else{
                        peek = head.item;
                        }
            return peek;
             }
}
```