**QUESTION 1**

1 points Save Answer

The behavior of adding and removing elements in stacks is:

- FIFO
- FIFA
- UFO
- LIFO

**QUESTION 2**

2 points Save Answer

The result of evaluating the postfix expression: 2 3 4 + 1 * -

- -5
- 7
- None
- 5

**QUESTION 3**

1 points Save Answer

What does this method do?

```
public static <T> void method2(Queue<T> q, T elem) {
     if (q.length() > 0) {
         int i = 0;
         int n = q.length();
         while(i < n) {
                T temp = q.serve();
                if (!elem.equals(temp))
                      q.enqueue(temp);
                i++;
         }
        }
}
```

- Infinite loop
- Nothing will change
- Remove all elements equal to elem from q
- Remove all elements not equal to elem from q

**QUESTION 4**

1 points Save Answer

The behavior of adding and removing elements in queue is:

- FIFO
- UFO
- LIFO
- FIFA

**QUESTION 5**

1 points Save Answer

The Big-Oh for the methods *push*, *pop*, *empty*, and *full* of the Stack for both implementations (Linked-List/Array) is:

- All O(1)
- All O(n)
- push/pop O(N), empty/full O(1)
- None

**QUESTION 6**

1 points Save Answer

A data structure in which elements can be inserted or deleted at/from both the ends but not in the middle is

- queue
- stack
- deque
- None

**QUESTION 7**

1 points Save Answer

The expected behavior of this method is to:

```
public static void method1(Stack s) {
    Stack s1 = new LinkedStack ();
    Stack s2 = new LinkedStack ();
    while(!s.empty())
        s1.push(s.pop());
    while(!s1.empty())
        s2.push(s1.pop());
    while(!s2.empty())
        s.push(s2.pop());
}
```

- Keep the bottom half of the stack s
- Keep the top half of the stack s
- Not change anything in stack s
- Reverse the stack s

**QUESTION 8**

**2 points** Save Answer

For a method in ArrayQueue - **circular array** -. replace the highlighted parts by the correct one for the method print. The method should print all elements in the queue.

```
public void print(){
    [          ] temp = head;
    for(int i = 0; i < [          ]; i++)
    {
        System.out.prinln([          ]);
        temp = [          ];
    }
}
```

**QUESTION 9**

**1 points** Save Answer

Suppose we have a **circular array** implementation of the queue class, with **8 elements** in the queue stored at **nodes[4]** (least recent element ) through **nodes[11]** (most recent element). **The Maximum Size is 15**. Where does the enqueue member function place the new element in the array?

- nodes[15]
- nodes[12]
- nodes[14]
- nodes[11]

**QUESTION 11**

**1 points** Save Answer

The expected behavior of this method is to:

```
public static void method2(Stack s) {
    Stack s1 = new LinkedStack ();
    int n = 0;
    while(!s.empty()) {
        s1.push(s.pop());
        n++;
    }
    n = n / 2;
    for(int i = 0; i < n; i++)
        s.push(s1.pop());
}
```

- Keep the bottom half of the stack s
- Keep the top half the stack s
- Reverse the stack s
- Not change anything in stack s

**QUESTION 10**

1 points Save Answer

The Big-Oh method enqueue a node-based on priority in a priority queue is:

- O(1)
- O(n)
- O(logn)
- $O(n^2)$

**QUESTION 12**

1 points Save Answer

What is the time complexity of the inserting element on the front of the **deque** implemented with a Circular Array?

- O(1)
- O(n)
- $O(n^2)$
- O(logn)

**QUESTION 13**

1 points Save Answer

As a user of the Stack ADT, consider the recursive method insertAtBottom, that takes a stack st and an element e, and insert element e at the bottom of the stack. You are not allowed to use any auxiliary data structures. Complete the code below by choosing the correct answer:

```
1. public static void insertAtBottom (LinkedStack st, T e) {
2.     if(...)
3.         ...
4.     else {
5.         ...
6.         ...
7.         ...
8.     }
9. }
```

Line 2:

- top == null
- None
- st.length() == 0
- !st.full()
- st.empty()

**QUESTION 14**

1 points Save Answer

As a user of the Stack ADT, consider the recursive method insertAtBottom, that takes a stack st and an element e, and insert element e at the bottom of the stack. You are not allowed to use any auxiliary data structures. Complete the code below by choosing the correct answer:

```
1. public static void insertAtBottom (LinkedStack st, T e) {
2.     if(...)
3.         ...
4.     else {
5.         ...
6.         ...
7.         ...
8.     }
9. }
```

Line 3:

- None
- top.data = e;
- insertAtBottom(st, e);
- st.push(e);
- return;

**QUESTION 15**

1 points Save Answer

As a user of the Stack ADT, consider the recursive method insertAtBottom, that takes a stack st and an element e, and insert element e at the bottom of the stack. You are not allowed to use any auxiliary data structures. Complete the code below by choosing the correct answer:

```
1. public static void insertAtBottom (LinkedStack st, T e) {
2.     if(...)
3.         ...
4.     else {
5.         ...
6.         ...
7.         ...
8.     }
9. }
```

Line 5:

- T tmp = st.pop();
- T tmp = top.data;
- None
- st.push(e);
- insertAtBottom(st, e);

**QUESTION 16**

1 points Save Answer

As a user of the Stack ADT, consider the recursive method insertAtBottom, that takes a stack st and an element e, and insert element e at the bottom of the stack. You are not allowed to use any auxiliary data structures. Complete the code below by choosing the correct answer:

```
1. public static void insertAtBottom (LinkedStack st, T e) {
2.     if(...)
3.         ...
4.     else {
5.         ...
6.         ...
7.         ...
8.     }
9. }
```

Line 6:

- insertAtBottom(st, e);
- st.push(e);
- T tmp = st.pop();
- insertAtBottom(st, tmp);
- None

**QUESTION 17**

1 points Save Answer

As a user of the Stack ADT, consider the recursive method insertAtBottom, that takes a stack st and an element e, and insert element e at the bottom of the stack. You are not allowed to use any auxiliary data structures. Complete the code below by choosing the correct answer:

```
1. public static void insertAtBottom (LinkedStack st, T e) {
2.     if(...)
3.         ...
4.     else {
5.         ...
6.         ...
7.         ...
8.     }
9. }
```

Line 7:

- st.push(e);
- top.data = tmp;
- None
- st.push(tmp);
- return;

**QUESTION 18**

1 points Save Answer

As a user of the Stack ADT, consider the static method *print*, which takes a stack *s* containing data of type String, and prints its elements from top to bottom. Stack *s* should be unchanged after the method is done. Complete the code below by choosing the correct answer:

```
1. public static void print(Stack<String> s) {
2.     ...
3.     ... {
4.         ...
5.         ...
6.         System.out.println(e);
7.     }
8.     ...
9.     ...
10.}
```

Line 2:

- List <String> l = new List <String>();
- Node<String> tmp = s.top;
- Stack <String> s2 = new Stack<String>();
- None
- Queue<String> q = new Queue<String>();

**QUESTION 19**

1 points Save Answer

As a user of the Stack ADT, consider the static method *print*, which takes a stack *s* containing data of type String, and prints its elements from top to bottom. Stack *s* should be unchanged after the method is done. Complete the code below by choosing the correct answer:

```
1. public static void print(Stack<String> s) {
2.     ...
3.     ... {
4.         ...
5.         ...
6.         System.out.println(e);
7.     }
8.     ...
9.     ...
10.}
```

Line 3:

- while(!s.empty())
- while(tmp != null)
- for(int i = 0; i < s.length(); i++)
- None
- while(!s.last())

**QUESTION 20**

**1 points** Save Answer

As a user of the Stack ADT, consider the static method *print*, which takes a stack *s* containing data of type String, and prints its elements from top to bottom. Stack *s* should be unchanged after the method is done. Complete the code below by choosing the correct answer:

```
1. public static void print(Stack<String> s) {
2.     ...
3.     ... {
4.         ...
5.         ...
6.         System.out.println(e);
7.     }
8.     ...
9.     ...
10.}
```

Line 4:

- String e = s.getData();
- String e = tmp.data;
- String e = s.retrieve();
- String e = s.pop();
- None

**QUESTION 21**

**1 points** Save Answer

As a user of the Stack ADT, consider the static method *print*, which takes a stack *s* containing data of type String, and prints its elements from top to bottom. Stack *s* should be unchanged after the method is done. Complete the code below by choosing the correct answer:

```
1. public static void print(Stack<String> s) {
2.     ...
3.     ... {
4.         ...
5.         ...
6.         System.out.println(e);
7.     }
8.     ...
9.     ...
10.}
```

Line 5:

- None
- s2.push(e);
- tmp = tmp.next;
- q.enqueue(e);
- l.insert(e);

## QUESTION 22

1 points Save Answer

As a user of the Stack ADT, consider the static method *print*, which takes a stack *s* containing data of type String, and prints its elements from top to bottom. Stack *s* should be unchanged after the method is done. Complete the code below by choosing the correct answer:

```
1. public static void print(Stack<String> s) {
2.     ...
3.     ... {
4.         ...
5.         ...
6.         System.out.println(e);
7.     }
8.     ...
9.     ...
10.}
```

Line 8:

- tmp = top;
- while(!s2.empty())
- None
- while(!l.last())
- for(int i = 0; i < q.length(); i++)

## QUESTION 23

1 points Save Answer

As a user of the Stack ADT, consider the static method *print*, which takes a stack *s* containing data of type String, and prints its elements from top to bottom. Stack *s* should be unchanged after the method is done. Complete the code below by choosing the correct answer:

```
1. public static void print(Stack<String> s) {
2.     ...
3.     ... {
4.         ...
5.         ...
6.         System.out.println(e);
7.     }
8.     ...
9.     ...
10.}
```

Line 9:

- s.push(s2.pop());
- None
- s.push(q.serve());
- top.next = tmp;
- s.push(l.retrieve());