Data Structure and Algorithm Analysis

2017-18 Mid-term Examination B

一、Fill caps

1、 array

2、 342*+3*63/-

3、 45

4、 queue、stack

5、 SSSXSXXX

6、 4

7、 0，3

二、Simple answers

1、 n/2

2、 -1 0 0 1 2 3 4

3、 public class DoublyLinkedList<E>{

int size;

Node<E> first;

Node<E> last;

private static class Node<E> {

E item;

Node<E> next;

Node<E> prev;

}

public add(int index, E element){

```
if (!(index >= 0 && index <= size)) {
    throw new IndexOutOfBoundsException("Index: " + index + ", Size:
" + size);
}
if (index == size) {
   final Node<E> l = last;
   final Node<E> newNode = new Node<>(l, element, null);
```

```
                last = newNode;
                if (l == null)
                    first = newNode;
                else
                    l.next = newNode;
                size++;
            } else {
                 Node<E> succ = node(index);
                 final Node<E> pred = succ.prev;
                 final Node<E> newNode = new Node<>(pred, element, succ);
                 succ.prev = newNode;
                 if (pred == null)
                     first = newNode;
                 else
                     pred.next = newNode;
                 size++;
             }


            }
            public delete(int index){
             if (!(index >= 0 && index < size)) {
             throw new IndexOutOfBoundsException("Index: " + index + ", Size: " +
size);
         }
         Node<E> x = node(index);
        final E element = x.item;
        final Node<E> next = x.next;
        final Node<E> prev = x.prev;

        if (prev == null) {
            first = next;
        } else {
            prev.next = next;
```

```
            x.prev = null;
        }

        if (next == null) {
            last = prev;
        } else {
            next.prev = prev;
            x.next = null;
        }

        x.item = null;
        size--;
        return element;

         }
        }
```

4、
```
        for(int i=0; i<n/2; i++){
            if (s[i] != s[n-i]){
                return False;
            }
        }
        return True;
```

三、Algorithm

1、Factorial

```
int factorial(n){
    if(n==1){
        return 1;
    }
    return n* factorial(n-1);
}
```

2、Firstly, try to find out how long the two linked lists, for example, one linked list is n, the other is m, suppose n >=m;

Secondly, search the longer linked list, after finding (n-m) elements, should search the two linked list together, when the two pointer point to the same node, this node is the first common node.

3、Postfix notation is easier to calculator than other notation. With two simple operations, the stack push and the stack pop can handle any normal expression. The operations are as follows:

If the current character is a digital, then push to stack,

if it is an operator, will be the top of the stack of two elements for the corresponding pop-up operation, the result reentry to stack.

When all the character operated on above way, the stack should be contain only one element, that is the final result.

4、Binary search

The maximum value possible: mvp = (xn – x1)/(m - 1)

Binary search form 0 to mvp(begin = 0, end = mvp), each time check whether the distance can be satisfied.

```
current value = 0
if check return true
   if mid is bigger than current value
        update the current value
   adjust the range from mid+1 to end
else if check return false
     adjust the range from begin to mid-1
```