Informatics 2, 3rd midterm (2018-05-14)

1	2	3	4	5	$\sum$

The answers should fit next to the questions, if you used a separate paper let us know clearly!

1. You can see an implementation of a binary tree. Implement a count and depth method which return the number of elements in the tree and the depth (height) of the tree. *(4 points)*

```
class Node(object):
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

    def insert(self, data):
        if self.data > data:
            if self.left is None:
                self.left = Node(data)
            else:
                self.left.insert(data)
        elif self.data < data:
            if self.right is None:
                self.right = Node(data)
            else:
                self.right.insert(data)
```

2. Theoretical questions *(4 points)*

a) How to search in a sorted list with binary search? Write the algorithm with your own words.

b) How to solve the tower of Hanoi with a recursive algorithm? Write the algorithm with your own words.

c) What is the difference between the dynamic programming and recursion?

d) When is a binary tree *unbalanced*? Give an example.

3. There is a text where everything between quotion marks (") is a comment. Write a function which returns a string with the comments removed. *(4 points)*

```
def erasecomment(text):
```

Example

```
>>> print erasecomment('cat "dog puppy" python')
cat  python
```

4. Finish the implementation if the "paint bucket" function. *(4 points)*

Suppose that the picture is 20 × 20 in size and contains only # and . characters.

```
picture = []
with open('picture.txt') as f:
    for line in f:
        picture.append(list(line.strip()))

def fill(x, y):
```

```
fill(0,0)
for x in picture:
    print ''.join(x)
```

5. The code on the right hand side is the calculator example. Write a representation function for printing the expression. Build the string from the expression tree recursively. *(4 points)*

Example:

```
print Node("2*(3+4)")
(2)*((3)+(4))
```

```
class Node(object):
    def __init__(self, kappa):
        i = -1
        if kappa.find("+") != -1:
            i = kappa.find("+")
        elif kappa.find("-") != -1:
            i = kappa.find("-")
        elif kappa.find("*") != -1:
            i = kappa.find("*")
        elif kappa.find("/") != -1:
            i = kappa.find("/")
        if i != -1:
            self.data = kappa[i]
            self.left = Node(kappa[:i])
            self.right = Node(kappa[i + 1:])
        else:
            self.data = float(kappa)
            self.left = None
            self.right = None

    def __repr__(self):
```