

# CS532 Homework 8

## Archana Machireddy

### Question 1

```
def print_tree(self, indent=""):
    print("%s%s"%(indent, self.value))
    next = self.child
    while next is not None:
        ret = next.print_tree(indent+" "*(len(self.value)))
        if ret is not None:
            return ret
        next = next.sibling
    return None
```

### Question 2

```
def split_node(self, prefix):
    b = self.value[len(prefix):]
    self.value = prefix
    c = self.child
    self.child = chars(b)
    self.child.parent = self
    self.child.child = c
    if c != None:
        self.child.child.parent = self.child
        next = self.child.child.sibling
        while next is not None:
            next.parent = self.child
            next = next.sibling
```

### Question 3

```
def find_partial(self, string, indent=""):
    if string.find(self.value) != 0:
        return (None, string)
    string = string[len(self.value):]
    last = self
    if string == "":
```

```

        return (self,string)
    next = self.child
    while next is not None:
        ret,sub = next.find_partial(string,indent+" ")
        if ret is not None:
            return (ret,sub)
        next = next.sibling
    return (last,string)

```

## Question 4

```

def add_word(self,word):
    node, sub = self.find_partial(word)
    next = node.child
    while next is not None:
        com = self.common_sub(next.value,sub)
        if com != "":
            next.split_node(com)
            if sub[len(com):] != "":
                next.add_child(chars(sub[len(com):]))
                self.set_word(next,sub[len(com):])
            else:
                next.word = True
            return
        else:
            next = next.sibling
    if sub != "":
        node.add_child(chars(sub))
        self.set_word(node,sub)
    return

def set_word(self,next,sub_added):
    node1 = next.child
    while node1 is not None:
        if node1.value == sub_added:
            node1.word = True
            break
        else:
            node1 = node1.sibling

def print_tree(self,indent=""):
    if self.word == False:
        print("%s%s"%(indent,self.value))
    else:
        print("%s%s W"%(indent,self.value))
    next = self.child

```

```

while next is not None:
    ret = next.print_tree(indent+" "*(len(self.value)))
    if ret is not None:
        return ret
    next = next.sibling
return None

def split_node(self,prefix):
    b = self.value[len(prefix):]
    self.value = prefix
    c = self.child
    self.child = chars(b)
    self.child.parent = self
    if self.word == True:
        self.child.word = True
    self.child.child = c
    if b != "":
        self.word = False
    if c != None:
        self.child.child.parent = self.child
        next = self.child.child.sibling
        while next is not None:
            next.parent = self.child
            next = next.sibling

def __init__(self,value):
    self.parent = None
    self.child = None
    self.sibling = None
    self.value = value
    self.word = False

```

## Question 5

```

def create_trie():
    l = lexicon()
    start = ""
    top = chars("")
    while True:
        words = l.next5(start)
        if len(words) < 1:
            break
        for w in words:
            top.add_word(w)
            start = words[-1]
    return top

```

```

def tokenize(trie,string):
    queue = [(0,[])]
    str_len = len(string)
    answers = []
    while queue:
        start, tokens = queue.pop(0)
        print(start,tokens)
        if start == str_len:
            answers.append(tokens)
        else:
            node, rem = trie.find_partial(string[start:])
            done = len(string)-len(rem)
            found = string[start:done]
            if tokens == []:
                new_tokens=[found]
            else:
                new_tokens = tokens+[found]
            if node.word == True:
                queue.append((done,new_tokens))
            print('Found word %s'%(found))
            print('Adding to frontier %s %i'%(found,done))
            next = node.parent
            l = len(node.value)
            while next is not None:
                new_l = len(found)-l
                new_ll = done - l
                if next.word == True:
                    if tokens == []:
                        new_tokens=[found[0:new_l]]
                    else:
                        new_tokens = tokens + [found[0:new_l]]
                    queue.append((new_ll,new_tokens))
                    print('Adding to frontier %s
%i'%(found[0:new_l],new_ll))
                else:
                    print(' Node %s is not a word'%(found[0:new_l]))
                    l = l+len(next.value)
                next = next.parent
    return answers

```