Q6:

a)



b)



c)

The algorithm first search the position of x, then recursively add strings, which are (l – x.length) or less levels below that position, to a list. Lastly, return the list.

Searching x takes time where is the length of x;

Add string to lists takes time where is the length of x, is the given length.

//returns a pointer to the node containing x(without $)

Node\* search(T, x){

p : pointer to a node

p = T.root;

for (auto& one\_char: x)

if (one\_char == 0){

p = p.left;

} else if (one\_char == 1){

p = p.right;

}

}

return p;

}

// add strings starting from pointer p with length less than or equal

// to "length" to my\_list

void add\_strings(old\_str, my\_list, p, length){

old\_str: string containing previous result

my\_list: list of strings as result

p : pointer to a node

length: string length

if (length >= 0){

if (p.is\_end()){

my\_list.add(old\_str + p.value);

}

add\_strings(old\_str + p.value, my\_list, p.left, length - 1);

add\_strings(old\_str + p.value, my\_list, p.right, length - 1);

}

}

List<string> Look(T, x, l){

T: Triee

x: given string

l: given maximum length

p : pointer to a node

p = search(T, x);

length = l - x.length();

my\_list : list of strings, initially empty

add\_strings(x, my\_list, p, length);

return my\_list;

}