Tutorial 5

1] i) Start
ii) Make a stack, stack().
iii) Make a function push(input) to push an element into the stack().
iv) Make a function pop() to pop top element of the stack().
v) Take an input number, n.
vi) Make a recursive function for factorial of n as factorial(n)

```
factorial(n)
{
    push(n);
    pop;
    if (n == 0)
        push(1);
    else
        push(n * factorial(n-1));
}
```

vii) Display the top element of the stack, i.e., factorial of n.
viii) Stop.

Dry run (for 4)

i) Start
ii) n = 4
iii) push (4)
pop
push 4 * factorial (3)

push (3)
pop
push 3* factorial (2)

push (2)
pop
push 2* factorial (1)

push(1)
pop
push 1* factorial (0)

push (0)
pop
push 1

Final element pushed at top will be 1 * 1 * 2 * 3 * 4 = 24
iv) Display stack[top], i.e., 24
v) Stop.

2] i) Start
ii) Create an array, arr [N] for characters.
iii) Create 2 more arrays, as left [N] & right [N].
iv) Traverse through arr [] and copy '(', '{' & '[' in left [] accordingly. Similarly, ')', '}' & ']' in right [].
v) Define a loop for N/2 and repeat the following:
for left [i], search the corresponding bracket starting from (n-i) in right [] and swap it with right [n-i].
vi) Copy (by replacing) left [] to arr [] and the for further half, right [] to arr [].
vii) Display arr [].
viii) Stop.

Dry run for {[()]}

i) arr ['{', '[', '(', ')', ']', '}']
ii) left ['{', '[', '(']
right [')', ']', '}']
iii) for left [0], right [2] is matched.
for left [1], right [1] is matched.
for left [2], right [0] is matched.
iv) arr ['{', '[', '(', ')', ']', '}']
v) {[()]}

⇒ for {{[}]}

i) arr ['{', '{', '[', '}', ']', '}']
left ['{', '{', '[']
right ['}', ']', '}']
iii) for left [0], right [2] matches.
for left [1], right [1] does not matches, right [0] matches, so swap them.
for left [2], right [0] matches.
iv) right [']', '}', '}']
arr ['{', '{', '[', ']', '}', '}']
v) {{[]}}