**PRACTICE 2**

**1. Algorithms:**

Status infix2postfix (String suf, String in) :

ret = TRUE

s = stack\_ini ()

while in[i] <> EoS :

// if the symbol being read is an operator

if isOperator (in[i]) == TRUE :

while stack\_isEmpty (s) == FALSE AND prec (stack\_top (s)) > prec (in[i]) :

ele = stack\_pop (s)

suf [j] = ele

j = j + 1

stack\_push (stack, in[i])

// else if the symbol being read is a left parenthesis

else if in[i] == '('

stack\_push (stack, in[i])

// else if the symbol being read is a right parenthesis

else if in[i] == ')' :

while stack\_isEmpty (s) == FALSE AND stack\_top (stack) <> '(' :

ele = stack\_pop (stack)

suf [j] = ele

j = j + 1

// pop the open parenthesis

stack\_pop (stack)

// else if the symbol being read is an operand, put it in suffix expression else :

suf[j] = in[i]

j = j + 1

i = i + 1

while stack\_isEmpty (s) == FALSE :

ele = stack\_pop (stack)

suf [j] = ele

j = j + 1

stack\_free (s)

return TRUE

**2. Hidden types:**

We could count the number of elements there are in a stack by popping all its elements, storing them in an auxiliar stack and after counting them, pushing the elements from the auxiliar stack to the main stack.

int main(){

Stack \*s1 = NULL, \*s2 = NULL;

Element \*ele = NULL;

int cont = 0;

s1 = stack\_init();

if (!s1) {

return 1;

}

s2 = stack\_init();

if (!s2) {

stack\_free(s1);

return 1;

}

for(i = 0; i < 5; i++){

ele = element\_init();

if (!ele) {

stack\_free(s1);

stack\_free(s2);

return 1;

}

if(element\_setInfo(ele, &i) == ERROR){

stack\_free(s1);

stack\_free(s2);

element\_free(ele);

return 1;

}

if(stack\_push(s1, ele) == ERROR){

stack\_free(s1);

stack\_free(s2);

element\_free(ele);

return 1;

}

element\_free(ele);

}

while (stack\_isEmpty(s1) == FALSE){

ele = stack\_pop(s1);

if(!ele){

stack\_free(s1);

stack\_free(s2);

return 1;

}

if(stack\_push(s2, ele) == ERROR){

stack\_free(s1);

stack\_free(s2);

element\_free(ele);

return 1;

}

element\_free(ele);

cont++;

}

while (stack\_isEmpty(s2) == FALSE){

ele = stack\_pop(s2);

if(!ele){

stack\_free(s1);

stack\_free(s2);

element\_free(ele);

return 1;

}

if(stack\_push(s1, ele) == ERROR){

stack\_free(s1);

stack\_free(s2);

element\_free(ele);

return 1;

}

element\_free(ele);

}

printf (“Stack size: %i”, cont);

stack\_free(s1);

stack\_free(s2);

element\_free(ele);

return 0;

}

**3. Design:**

We would not be able to do it because the element structure has a pointer to a char, not to Node\* or void\*.