Page No

Date

Devanshu Surana
1032210755, PC-12
FDS Lab Assignment 7

Problem statement:
Implement stack as an ADT and apply it for different expression conversions (infix to postfix or infix to prefix (Any One), prefix to postfix or postfix to infix).

Objective
1. To study stack and its operations
2. To study the importance of expression conversions.

Theory:
Write in brief about stack and different Expression conversion.
- Stack: Special case of ordered list also called as restricted/controlled list where insertion and deletion happens at only one end called as top of stack.
- Elements are added to end removed from the top of the stack (the most recently added items are at the top of the stack).
- Infix expression: It is the general notation used for representing expression.
- In this expression the operator is fixed in between the operands.
  Ex. a + b * c
- Postfix expression: (Reserve Polish Notation)
- In this expression the operator is placed after the operands.
  eg: abc* +

Page No.
Date

- Prefix Expression
In this expression the operators are followed by operand i.e the operators
Ex : * + abc
- Operator Precedence governs evaluation order. An operator with higher precedence is applied before an operater with lower precedence.
Expression conversion forms

Infix	Postfix	Prefix
A + B	AB +	+ AB
(A+B)* (C+D)	AB+CD·+*	* + AB + CD
A - B / (C * D ^ E)	ABCDE ^ * / -	- A / B * C ^ DE

i) Infix to postfix
ii) Infix to prefix
iii) Prefix to postfix
iv) prefix to infix
v) postfix to prefix
vi) postfix to infix

Platform: 64-bit Open Source Linux or its derivatives
- Open source C programming tool like gcc/Eclipse Editor.

PSEUDO Code:
a) Push:

```
void push (pnode * top, element)
{
pnode * temp = (pnode) malloc (size of (pnode));
temp → item = item;
```

Page No
Date

```
temp → next = NULL;
if (top == NULL)
    top = temp;
else
    temp → next = top;
            top = temp.
}
```

b) Pop :

```
int pop (pnode * top)
{
  pnode * temp = top;
  element item;
  if (top == NULL)
     print (empty stack)
  else
     item = temp → item

     top = temp → next
     free (temp)
     return item;
}
```

c) Infix to postfix.

```
void in_post (inexp [])
{
  K = 0; i = 0;
  tkn = inexp [i];
  while (tkn != ~~#~~ '\0')
  {
    if (tkn is an operand)
```

```
        { postexp [k] = inexp [i];
          k++;
        }
       else
      { if (tkn == 'c' // open parenthesis
        { while [(tkn = pop()) != 'c')
          { postexp [K] = tkn; k++; }
        }
       else
      { while (stack not empty && isn (stk [top]) >=
                                         icp (tkn))
        { post exp [K] = pop(); k++; )
       push (tkn);
      }
     }
    }
    i++;
    tkn = inexp [i]; }
    while (stack not empty)
   { post exp [K] = pop();
     K++; }
   }.
```

d) postfix to infix:

```
    void post -in().
   { l = len (postfix []);
     for (i = 0   to  i-1) {
     tkn = next character in postfix [];
     if (tkn is an operand)
         push (tkn);
     else
```

```
        op2 = pop[];
        op1 = pop[];
        expr = ('C' op1, op2, +kn, ')')
        push (expr);
        }
    infix = pop ();
    }
```

Time Complexity.
1) Push = O(1)
2) Pop = O(1)
3) Infix to postfix = O(n)
4) Postfix to Infix = O(N)

Conclusion : Thus, implemented stack operation assignment using array concept.

FAQ's.

Ans 1] Much easier to translate to format that is suitable for direct execution.
They are entirely unombigious while infix notation requires precedence and associativity rules to dismobigious it.
Can be evaluated faster than infix expression

Ans 2] Evaluation of prefix/postfix expression is very easy. We will visit each element one by one: If the current element is an operand, we push it to stack. And if it is an operator, we will pop two operands, perform operation on them and push result into stack.

Page No. ____
Date ____

Ans 3] ICP and ISP are terms used in expression conversions, ICP means incoming prisrity. i.e priority of operator in expression which we are currently converting to certain expression type. While ISP means instack priority i.e priority of operator in stack.

Ans 4) Stack uses LIFO principle. As in expression conversions we constantly need to push and pop element which last entered the stack, using stack for expression conversion beome good choice.

Ans 5) Stack full condition → top = (maxsize of stack - 1)
Stack empty condition → top = [-1]

2/11/22